

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/>

LOUGHBOROUGH  
UNIVERSITY OF TECHNOLOGY  
LIBRARY

AUTHOR

MARTIN - SOLIS, G VOL 2.

COPY NO.

110623/02

VOL NO.

CLASS MARK

copy

0 FEB 1983  
LOAN 1 MTH + 2  
UNLESS RECALLED  
GAS CORR.

date due  
for return -  
24 DEC 1983  
LOAN 1 MTH + 2  
UNLESS RECALLED

~~Dec 15 1983~~  
57 NOV 1983  
LOAN 3 WKS. + 3  
UNLESS RECALLED  
LIBRARY USE  
ONLY.  
Date Due  
15 FEB 1990  
LOAN 3 WKS. + 3  
UNLESS RECALLED  
LIBRARY USE  
ONLY

011 0623 02



Loughborough University of Technology Library	
Date	Mar. 79
Class	
Acc. No.	110623 / 02

<del>date due</del>	LOAN C.	
<del>date due:-</del> <del>- 4 FEB 1991</del>	<del>date due:-</del> <del>- 9 SEP 1991</del>	
<del>LOAN 3 WKS. + 3</del> <del>UNLESS RECALLED</del> <del>GALWAY</del>	<del>LOAN 3 WKS. + 3</del> <del>UNLESS RECALLED</del>	
<del>date due:-</del> <del>12 JUN 1991</del>	<del>← 1 JUL 1994</del>	
<del>LOAN 3 WKS. + 3</del> <del>UNLESS RECALLED</del>		

## APPENDIX II

RTL/2 AND RSX DESCRIPTIONTable of Contents

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

## 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.

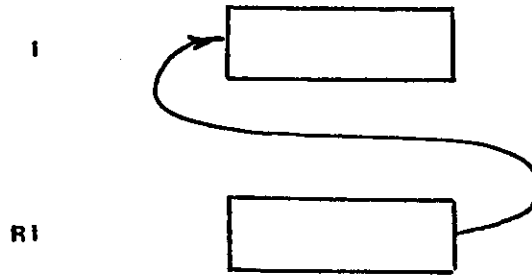
#### II.2.3.1 Reference modes

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:

```
INT I;
```

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

will declare an 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 AI. 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:

```
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:

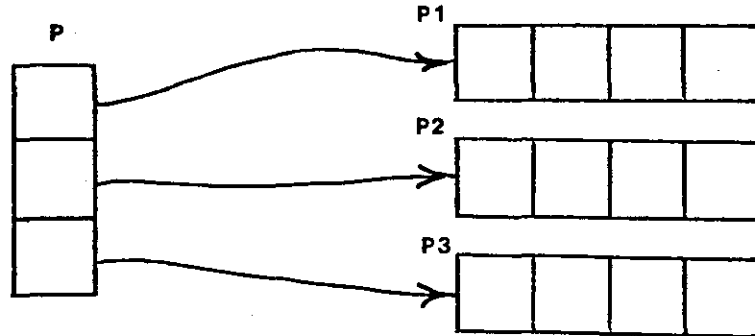
```
ARRAY(10) REAL A := (0.1,0.0(8),0.1);
```

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:

```
ARRAY(3) REF ARRAY INT P := (P1,P2,P3);
```

```
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 `ARRAY(3,4)INT P` will create the same structure except that the subarrays are now anonymous. Access to elements of `P` can be expressed as `P(I) (J)` or `P(I,J)` which are equivalent.

#### 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 `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. <sup>(5)</sup>

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

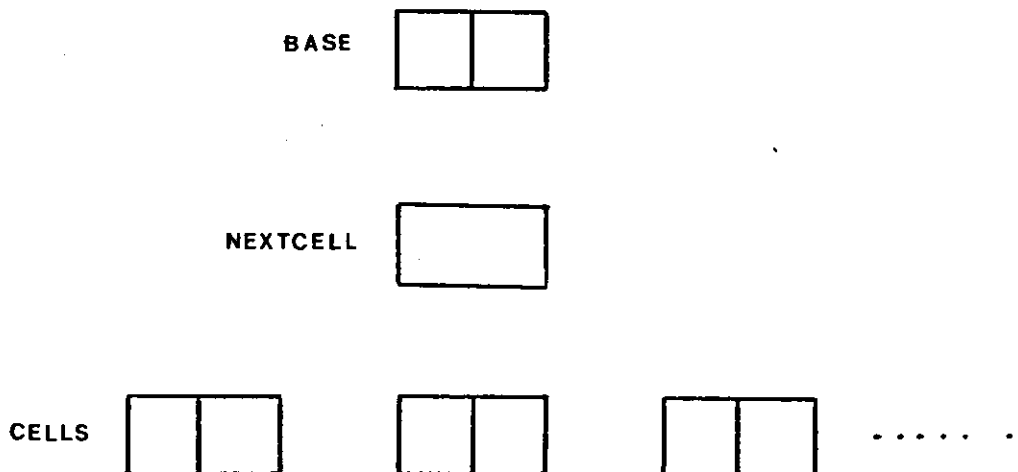
following:

```
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:





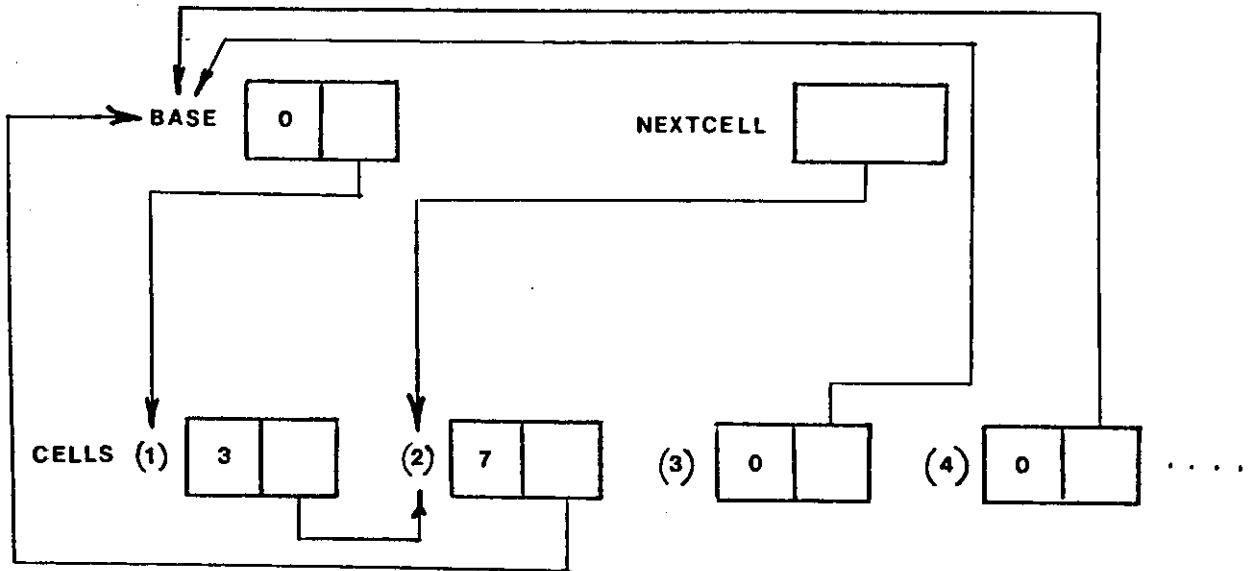
Initial values of records are indicated using a notation similar to that for arrays with the initial values of the components separated by commas and the whole in brackets. Unlike arrays repetition factors are not allowed. If the former example is initialised as follows:

```
CELL BASE: = (0, CELLS(1));
```

```
REF CELL NEXT CELL: = CELLS (2);
```

```
ARRAY(100)CELL CELLS: = ((3, CELLS(2), (7,BASE), (0,BASE) (98);
```

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:

```

BASE.VA
NEXTCELL.NEXT
CELLS(2).NEXT.VA

```

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.

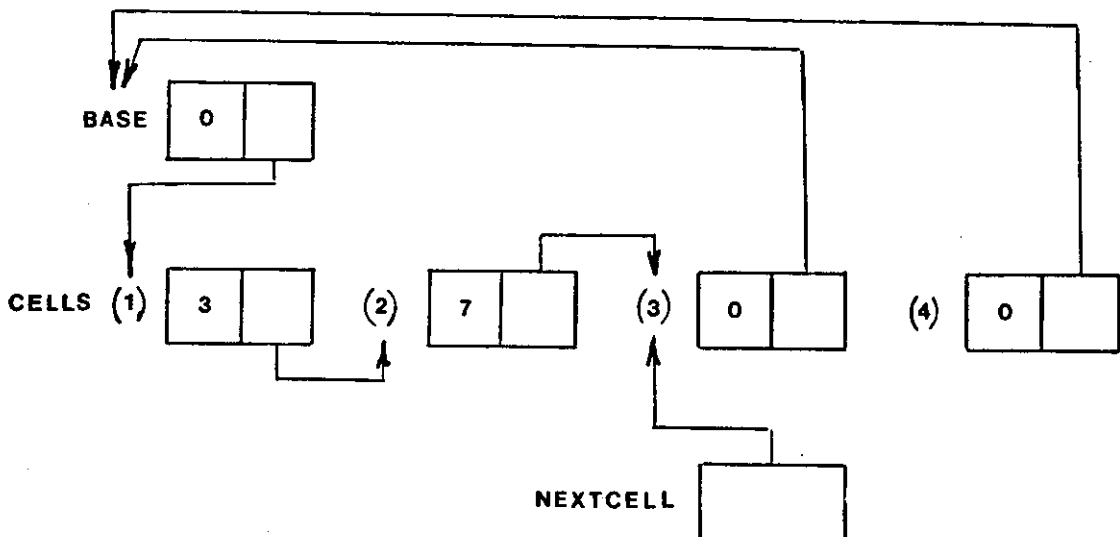
In the example used to illustrate the manipulation of records there are two declarations BASE and NEXTCELL that can be used as "pointers" to manipulate the list of CELLS. The pointers are used to "remember" a place in the list and to modify this list. In the example mentioned BASE is the pointer used 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:

```

NEXTCELL := CELLS(3)
CELLS(2).NEXT := NEXTCELL

```

The interlinked structure will look as follows:



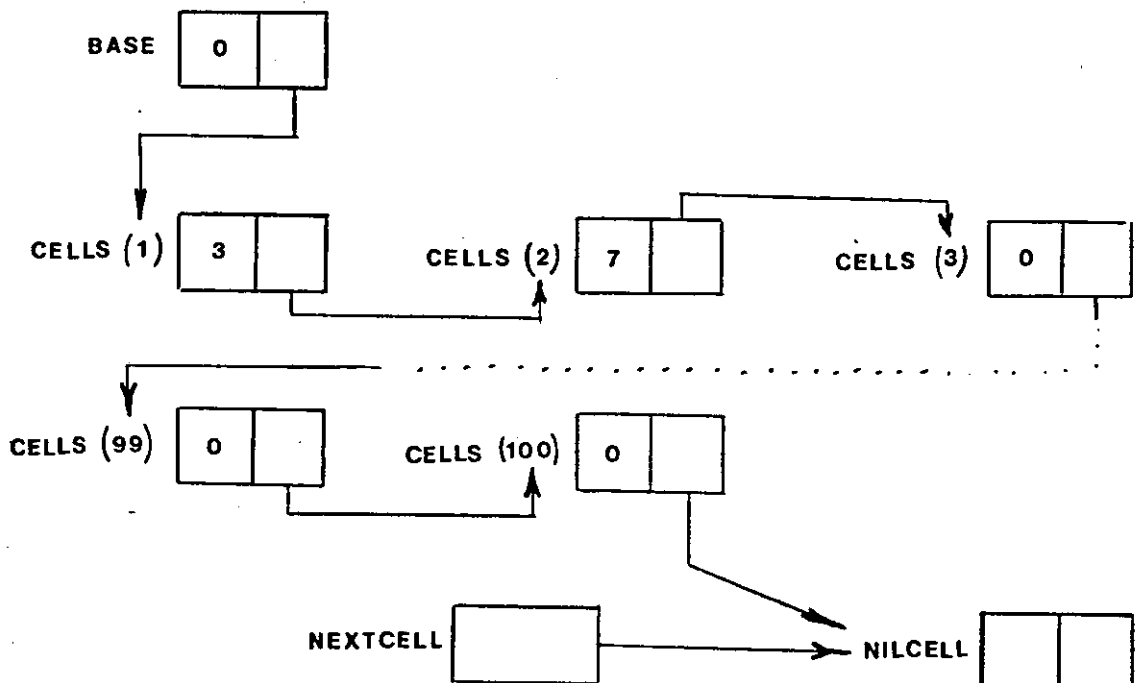
Sometimes in order to terminate the list the last element should point to no other element of the list. To solve the problem NILCELL is defined as follows:

```
CELL NILCELL := (0, 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:

```
NEXTCELL := NILCELL  
CELLS(100).NEXT := 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 semi-colon, a series of declarations separated by semi-colons 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:

```
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-11M 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-11M 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-11M 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-11M 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-11M 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-11M Executive. In both cases, the interrupts are channelled to the proper service routine by the hardware. The



combination of the PDP-11M hardware interrupt system and the RSX-11M software service routine provides the efficient interrupt processing required for effective real time operation.

### II.3.1 RSX-11M 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-11M 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-11M 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-11M 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-11M 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-11M 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-11M 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.<sup>(6)</sup>

#### 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.<sup>(15)</sup>

### 2) Compilation

A typical compiler command string would be:

```
RTL XAMPLE, TI:=XAMPLE/TI:XAMPLE/CS:XAMPLE
(1)   (2)   (3)   (4)       (5)       (6)
```

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
(1)  (2)  (3)
```

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.<sup>(13)</sup> 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=BA1,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 BA1.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-11M manual.<sup>(3)</sup>

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:

CAL, TI: /SH=BAL, CAL

T/LB, ELB/LB

/

UNITS=3

ACTFIL=2

ASG=TI:1:2:3

PAR=GEN:40000:40000

PRI=80

TASK=...CAL

COMMON=DATBAS:RW

LIBR=FTREES:RO

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.

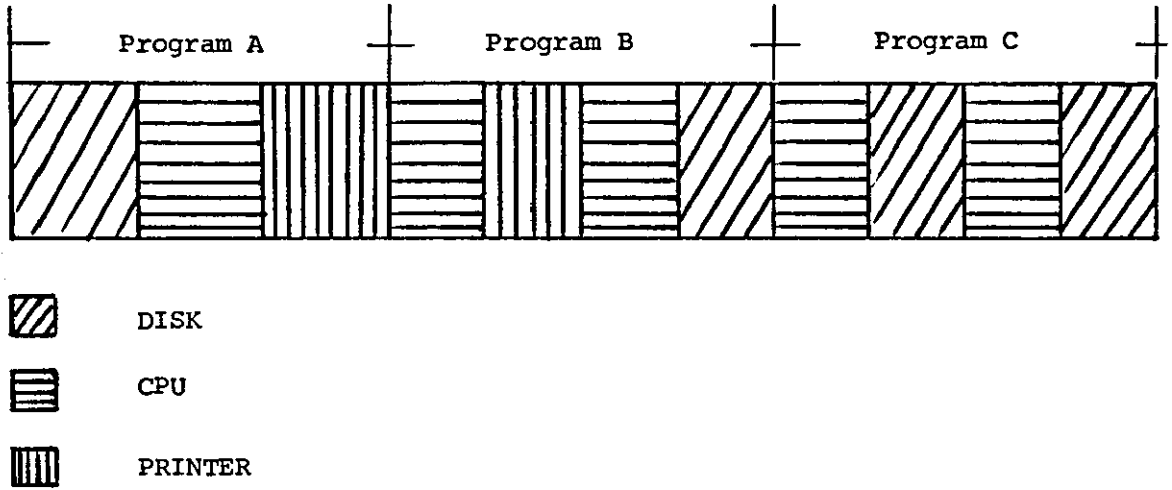


Fig. III.3.1 Sequential program execution

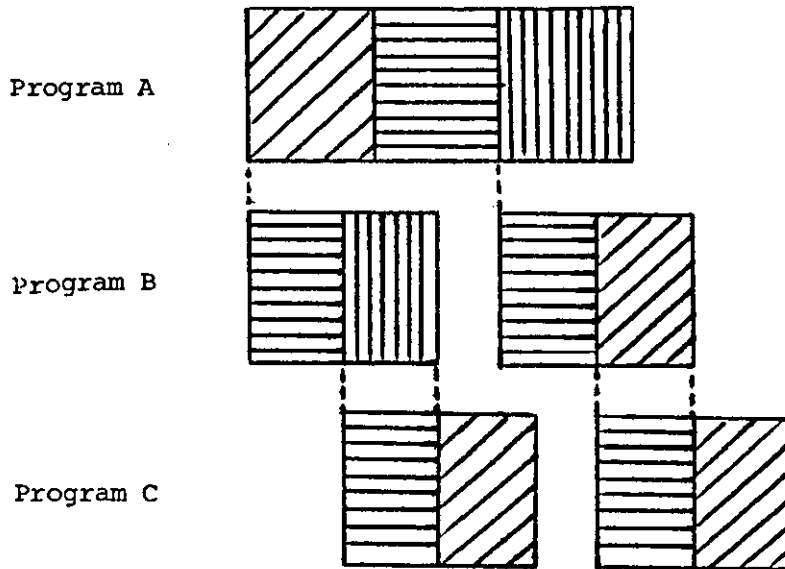


Fig. II.3.2 Concurrent program execution under  
RSX-11M

1. PREPARATION	2. COMPILATION	3. ASSEMBLY	4. LINKING
User creates the file XAMPLE.RTL by means of the RSX editor	User runs the RSX-11M task "RTL" to produce the MACRO assembly file	User runs the RSK-11M Task "MAC" to produce a binary file	User runs the RSX-11M Task Builder task "BTR" to produce the task image file.

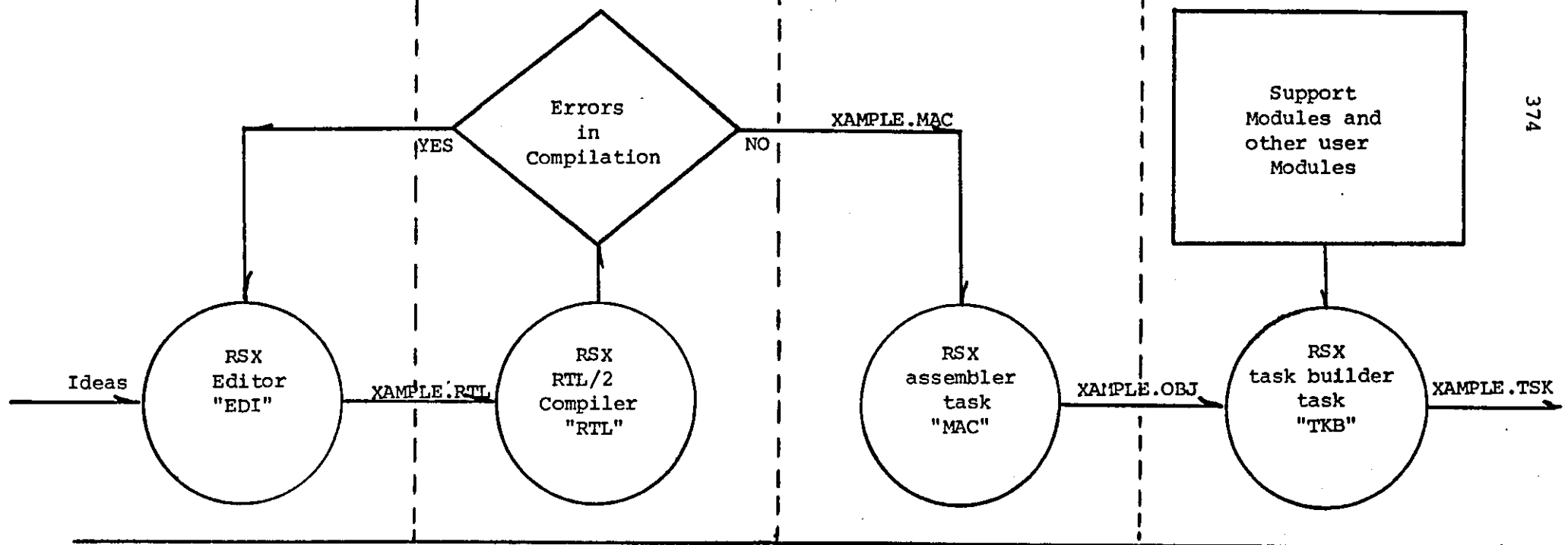


Fig. II.4.1 Steps involved in producing an RTL/2-written task for RSX-11M



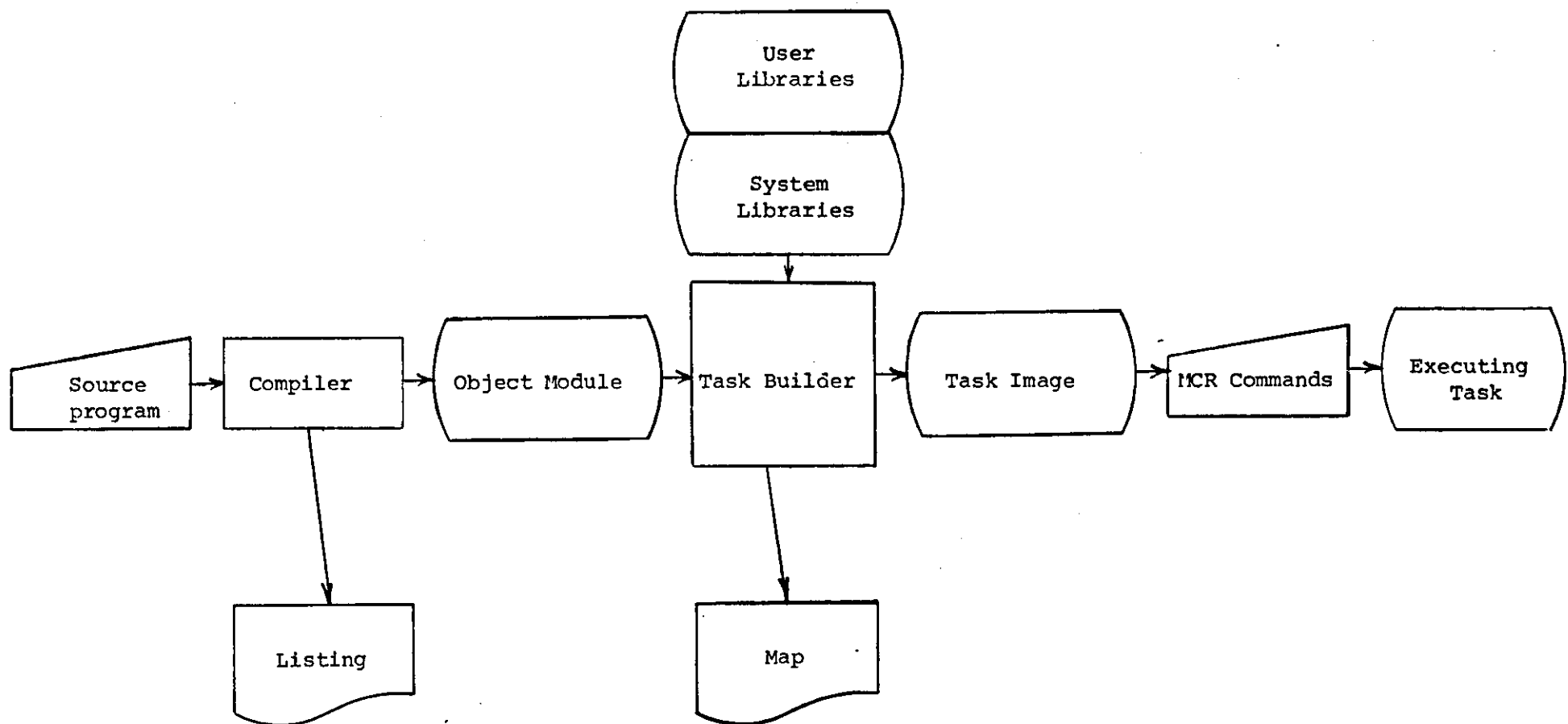


Fig. II.4.2 Preparing a source program for Execution

### References

- (1) ANDOW, P.K., "RTL/2 on the Chem. Eng. PDP-11/20 Computer", Loughborough Univ. of Technology; 1977.
- (2) BARNES, J.G.P., "RTL/2 Design and Philosophy"; Heyden & Son Ltd.; 1976.
- (3) DIMMER, C.I., STEVENSON, G.C. and MEREDITH, H.R., "DEC RSX-11, RTL/2 User Manual", SPL International; Version 2; London, 1975.
- (4) ECKHOUSE, R.H. Jr., "Minicomputer Systems: Organization and Programming (PDP-11)"; Prentice-Hall Inc.; Englewood Cliffs, New Jersey; 1975.
- (5) FOSTER, J.M., "List Processing"; MacDonald/Elsevier, London; 1967.

### RSX-11M Manuals:

- (6) "Executive Reference Manual"; (DEC-11-OMERA-B-D); Digital Equipment Corporation; Maynard Mass; 1975.
- (7) "Introduction to RSX-11M"; (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.
- (9) "I/O Operations Reference Manual"; (DEC-11-OMFSA-B-D); Digital Equipment Corporation; Maynard Mass; 1975.
- (10) "Macro-11 Reference Manual"; (DEC-11-OMMAA-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.
- (13) "Task Builder Reference Manual"; (DEC-11-OMTBA-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

- (15) "Language Specification"; RTL/2 ref. 1, ICI Ltd., 1974.
- (16) "Standard Stream I/O"; RTL/2 ref. 5, ICI Ltd., 1974.
- (17) "System Standards"; RTL/2 ref. 4; ICI Ltd., 1974.
- (18) "Training Manual"; RTL/2 ref. 3; ICI Ltd., 1974.

APPENDIX IIIPrograms and TasksTable of Contents

<u>Section</u>	<u>Page</u>
III.1 <u>Introduction</u> .. .. .	380
III.2 <u>Records and the Data Base</u> .. .. .	380
III.2.1 <u>Introduction</u> .. .. .	380
III.2.2 <u>Records</u> .. .. .	381
III.2.2.1 <u>MODE BOUNDECO</u> .. .. .	381
III.2.2.2 <u>MODE DATAV</u> .. .. .	382
III.2.2.3 <u>MODE EVENT</u> .. .. .	382
III.2.2.4 <u>MODE FAS</u> .. .. .	383
III.2.2.5 <u>MODE GATE</u> .. .. .	383
III.2.2.6 <u>MODE INDEX</u> .. .. .	383
III.2.2.7 <u>MODE STREAM</u> .. .. .	384
III.2.2.8 <u>MODE UNITDE</u> .. .. .	384
III.2.2.9 <u>MODE VARIAB</u> .. .. .	384
III.2.3 <u>Data Base</u> .. .. .	385
III.3 <u>Tasks</u> .. .. .	385
III.4 <u>Programs</u> .. .. .	388
<u>Tables</u>	
III.1 <u>Look-up Table for Faults</u> .. .. .	552
III.2 <u>Look up Table for Type of Units</u> . . . . .	553
III.3 <u>MODE BOUNDECO Description</u> .. .. .	554
III.4 <u>MODE DATAV Description</u> .. .. .	554
III.5 <u>MODE EVENT Description</u> .. .. .	555
III.6 <u>MODE GATE Description</u> .. .. .	556
III.7 <u>MODE INDEX Description</u> .. .. .	556
III.8 <u>MODE STREAM Description</u> .. .. .	557
III.9 <u>MODE UNITDE Description</u> .. .. .	557
III.10 <u>MODE VARIAB Description</u> .. .. .	558
III.11 <u>Description of the Pointers Defined in the</u> <u>Data Base</u> .. .. .	559
III.12 <u>Description of the Tasks</u> .. .. .	560
<u>Figures</u>	
III.2.3.1 <u>Data base program</u> .. .. .	561
III.2.3.2 <u>Memory layout used in this work</u> .. .. .	565
III.2.3.3 <u>List structure for topology of a Two pipe</u> <u>and valve system</u> .. .. .	566
III.3.1 <u>Task BMT indirect command file</u> . . . . .	567
III.3.2 <u>Task BTR indirect command file</u> . . . . .	568
III.3.3 <u>Task BTR overlay description</u> . . . . .	569

SectionPage

III.3.4	Task CAL indirect command file .. .. .	570
III.3.5	Task DEB indirect command file .. .. .	571
III.3.6	Task DEB overlay description .. .. .	572
III.3.7	Task DES indirect command file .. .. .	573
III.3.8	Task LIT indirect command file .. .. .	574
III.3.9	Task PMT indirect command file .. .. .	575
III.3.10	Task PRI indirect command file .. .. .	576
III.3.11	Task PTR indirect command file .. .. .	577
III.3.12	Task REU indirect command file .. .. .	578
III.3.13	Task RFA indirect command file .. .. .	579
III.3.14	Task RTU indirect command file .. .. .	580
III.3.15	Task RVA indirect command file .. .. .	581

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

### 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:

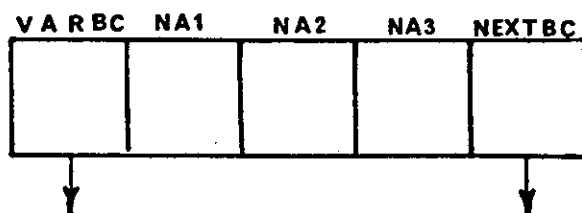
- 1) Use of 8 bit BYTES (rather than 16 bit INT's) to conserve storage space.
- 2) Use of bit manipulation to pack most of the information into BYTE or INT data.
- 3) Use of integers related to look-up tables instead of large repetitive strings.
- 4) Versatile records were preferred to specific ones.

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



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

HL	LL	PRVA	ACTVA	PRIO	PREX

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

IDEN	UNID	VAR	INO	NAVA	FAULT	NEXT	TOP	BACKG	NEXTG	UNNO	PROB
				↓		↓	↓	↓	↓	↓	

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.

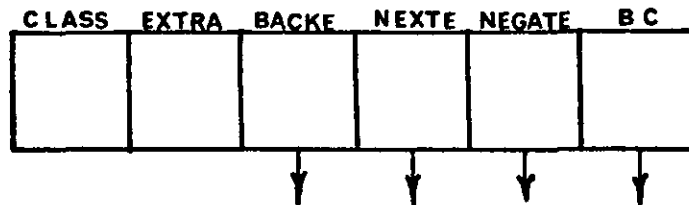


#### III.2.2.4 MODE FAS



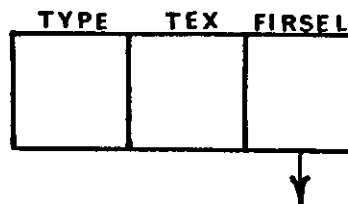
This mode is used to deal with long strings in the programs such as the name of the units or the name of faults. It is an array of bytes and has only one field L.

#### III.2.2.5 MODE GATE



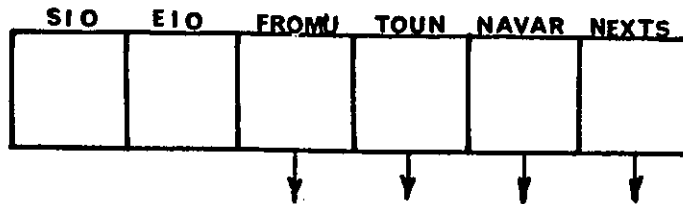
This mode is used to deal with the data related to the gates used in the trees and minitrees. The description of its fields is given in Table III.6

#### III.2.2.6 MODE INDEX



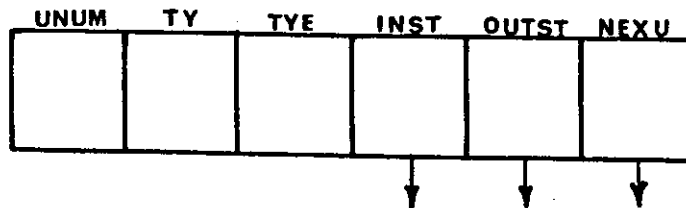
This mode is used to ease the location of the right minitree when the fault tree is being constructed. Its fields are described in Table III.7.

### III.2.2.7 MODE STREAM



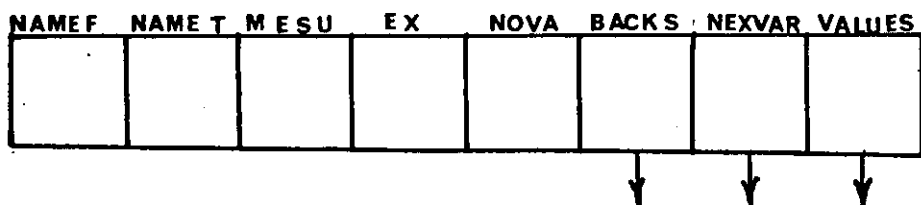
This mode is used to deal with the data of the unit's streams in the topology of the system under study. Its fields are described in Table III.8.

### 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 inter-linked 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 (DKØ:) 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  DKØ:A.TNT;1 = DKØ:PIPE.DAT
BTR  TI: = TI:*1
CAL  TI: = TI:*Ø
DEB  DKØ:VALVE.PRO;1 = TI:*3*3
DES  DKØ:B.TNT;1 = DKØ:HEXVAL.DAT
LIT  TI: = TI:
PMT  TI: = TI:
PRI  DKØ:HEXVAL.PRO;1 = TI:
PTR  TI: = TI:
REU  TI: = TI:
RFA  TI: = DKØ:FAULTS.DAT
TRU  TI: = DKØ: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 BUFLN = 72;
LET IOSERR = 5040;
LET IPUT = OCT 1000;
LET QIOERR = 5038;
LET WAIERR = 5039;

```

```

MODE BUFBLK (INT BDUMWRD,
             REF BYTE BBUFADD,
             INT BBUFSIZ, BVFCHAR, 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 INEXT := 0;
  IOSTAT IS := ('0', '0', 0);
  BUFBLK BB := (12, BUF(1), BUFLN, 0, 0, 0, 2);
  INT ITOTAL := 0;
ENDDATA;

```

```

EXT PROC () RRNUL;
EXT PROC (INT) RSXWTS, TESDSW;
EXT PROC (INT, INT, INT, INT, REF 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, IS, RRNUL, BB);
        TESDSW(QIOERR);

        RSXWTS(2);
        TESDSW(WAIERR);

        IF IS.IOSTLOW # 1
          THEN ERP(IOSERR);
        END;
        ITOTAL := IS.IOSTVAL + 1;
        INEXT := 1;

```

```

  ENDS;
  REP;
  RETURN(SP); % JUST TO SATISFY COMPILER %
ENDPROC;

```

# X ALTERNATIVE OUTPUT PROCEDURE X

```

LET BUFLN = 72;
LET IOSERR = 5017;
LET OPUT = OCT 400;
LET VFC = OCT 0;
LET QIOERR = 5016;
LET WAIERR = 5026;

```

```

MODE BUFBLK (INT BDUMWRD,
             REF BYTE BBUFADD,
             INT BBUFSIZ, BVFCHAR, 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 := ('E', 'E', 2);
  BUFBLK BB := (12, BUF (1), 1, VFC, 2, 2, 2);
ENDDATA;

```

```

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

```

```

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

```

```

  IF ONEXT = BUFLN OR B = CR OR B = VT
  THEN BB.BBUFSIZ := IF B = VT
                     THEN ONEXT - 1
                     ELSE ONEXT
                     END;

```

```

  ONEXT := 0;
  RSXQIO (OPUT, 2, 2, 2, IS, RRNUL, BB);
  TESDSW (QIOERR);

```

```

  RSXWTS (2);
  TESDSW (WAIERR);

```

```

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

```

```

END;
ONEXT := ONEXT + 1;
RETURN;
ENDPROC;

```



% DATA OF A 'B' EVENT %

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

ENT PROC BEVT(REF EVENT E);

  E.BACKG := GATEPO;

  L1:GETFRA(E.PROB,0.050,1,"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) SBITBY;
EXT PROC () INT RETYUN;
```

```
ENT 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,2,3,1,"0-OUT,1-IN,2-NO VAR.,3-1-VAR",L1);
        SBITBY(E.INOU,IN);
        E.FAULT := REFATR("++");
        TOTEV := TOTEV+1;
        IF TYEV = 'M'
          THEN IF I # CI
            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);
          ELSE TWRT("#NL#EVENT UNDEFINED"
            "(BMINT.RTL)");
        END;
        I := I+1;
      ENDBLOCK;
    GETBYT(TYEV,3,"TYPE OF EVENT(* TO TERMINATE INPUT)");
    REP;
    LOT(COLO).TYPE := BYTE(UN);
    LOT(COLO).FIRSEL := EVT(CI);
    COLO := COLO + 1;
    CI := CI+TOTEV;
    UN := RETYUN ();
  REP;
ENDPROC;
```

```

OPTION(1) BC,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) RSXOP1, RSXOP0;
EXT PROC(INT) INT RSXSW1, RSXSW0;
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 FILLAT;
  ARRAY(70) BYTE FILE := "#SP(70)#";
ENDDATA;

```

```

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

```

```

  TREAD(FILE, "=");
  RSXOP0(1, FILE);
  RSXSW0(1);

```

```

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

```

```

  TREAD(FILE, "#LF#");
  IN := ECHOIN;
  RSXOP1(2, FILE);
  RSXSW1(2);
  BMINT();
  RSXCL(1);
  RSXCL(2);

```

ENDPROC;

OPTION(1) BC,TR;

TITLE

CONSTRUCTION OF TREE;

LET COMERR = 5000;

LET SP = OCT 42;

LET LF = OCT 12;

EXT PROC(INT) BUILT;

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,ERRLUN;

INT RSXDSW;

ENDDATA;

DATA FILDAT;

ARRAY (35) BYTE IFILE := "#SP(35)#";

ARRAY (35) BYTE OFILE := "#SP(35)#";

ENDDATA;

ENT PROC RRJOB();

INT I;

ERRLUN := 3;

IN := COMIP;

IF IN() # 'B' OR IN() # 'T' OR IN() # 'R' OR IN() # ' ' THEN ERP(COMERR);

THEN ERP(COMERR);

END;

TREAD(OFILE,"=");

TREAD(IFILE,"\*");

I := IREAD();

IN := ALMIP;

OUT := ALMOP;

BUILT(I);

ENDPROC;

# 3 BUILDING THE EVENTS 3

```

EXT PROC () REF VARIAB CHEVAR;
EXT PROC () CHEEDA,CHECKD,NEW EVT;

ENT PROC BUILDE();
  IF FLAG = 0
    THEN EST := EST.NEXT;
    ELSE FLAG := 0;
  END;
BLOCK
  REF VARIAB V1 := CHEVAR();
  IF V1 ==: NILVAN 3NO SUCH VARIABLE OR END 3
    THEN CHECKD();
    RETURN;
  END;
  VST := V1;
  IF EST.IDEN = 'B'
    THEN NEW EVT();
    ELSE CHEEDA();
  END;
ENDBLOCK;
ENDPROC;

```

% BUILDING THE REST OF THE TREE %

EXT PROC () BUILDE;

ENT PROC BUILDR();

WHILE EAST.IDEN # 'M' DO

IF EAST.NEXTG :=: NILGA OR

EAST.NEXTG :=: QM

THEN IF EAST.NEXT :=: NILEV

THEN EAST := EAST.BACKG.BACKE;

GST := EAST.BACKG;

BCST := GST.BC;

UST := EAST.UNNO;

GOTO L1;

ELSE EST := EAST.NEXT;

UST := EAST.UNNO;

END;

FLAG := 1;

C := '\*';

BUILDE();

END;

L1:REP

ENDPROC;

\*\*\*\*\*  
.....  
% BUILDING THE TREE %

EXT PROC ( ) TOEVT, BUILDE, BUILDR, COPYGA;

ENT PROC BUILDT();

IF BT = 0

THEN TOEVT();

BUILDE();

ELSE COPYGA();

BUILDE();

BUILDR();

END;

ENDPROC;



\*\*\*\*\*  
X BUILT THE TREE X

EXT PROC ( ) SCANVA, DESIGN;

ENT PROC BUILTT(INT I);

RTD := 1;

IF RTD = 1 X REAL TIME ANALYSIS X

THEN SCANVA();

ELSEIF RTD = 0 X DESIGN ANALYSIS X

THEN DESIGN ( );

ELSE TWRT("#NL\*\*\*\* ERROR IN BUILTT.RTL");

END;

ENDPROC;

```
OPTION(1) BC,TR;
TITLE
CLEAR AND/OR LINK ARRAYS;
```

```
LET COMERR = 5000;
LET SP = OCT 40;
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,ERRLUN;
  INT RSXDSW;
ENDDATA;
```

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

```
ENT PROC RRJOB();
  INT I;
  ERRLUN := 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;
```

```
ENT 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;
```

402  
Z CHECK B. C. AND N. A. C. IN TREE Z

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 L1;

END;

G := E.BACKG;

REP;

RETURN (V);

END;

L1:N := EST.FAULT;

IF N = E.FAULT

THEN TWRT("#NL#EVENT ALREADY IN THE SAME BRANCH#CR#");

END;

IF N <= 16 THEN ANS := GETBIT(BCA.NA1,N);

ELSEIF N <= 32 THEN ANS := GETBIT(BCA.NA2,N-16);

ELSEIF N <= 48 THEN ANS := GETBIT(BCA.NA3,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 = SP

THEN RETURN (NILVAN)

END;

RETURN (CHEVAR());

END;

END;

RETURN(V);

ENDBLOCK;

ENDPROC;

2 CHECK DIAMOND EVENT 2

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

ENT PROC CHECKD();

IF GST.NEXTE ==: NILEV

THEN GST := DELGAT(EAST, GST);

BCST := GST.BC;

EAST.NEXTG := NILGA;

ELSE EAST.NEXT := NILEV;

END;

ENDPROC;

\* CHECK IF THE EVENT HAS BEEN DEVELOPED ALREADY \*

```
EXT PROC () NEW EVT;
EXT PROC (REF EVENT) NODEVL;
EXT PROC (REF EVENT) INT CHSEBC;
```

```
ENT 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 NODEVL(E);
            ELSE NEW EVT();
          END;
          RETURN;
        END;
      ENDBLOCK;
    END;
    I := I+1;
    E := EVT(I);
  REP;
  NEW EVT();
ENDPROC;
```

% CHECK 'EX-OR' GATE %

EXT PROC (BYTE,INT) INT GETBIB;

EXT PROC (REF VARIAB) REF VARIAB CHEBC;

ENT PROC CHEEOR (REF VARIAB V) REF VARIAB;

IF GETBIB(GST.CLASS,2) = 1

THEN IF GST.NEXTE :#: NILEV

THEN BLOCK

REF EVENT E := GST.NEXTE;

WHILE E.UNNO :#: NILUNI DO

IF E.NAVA.MESU = 'Y'

THEN

TWRT("#NL#VARIABLE ALREADY ON"  
"(EX-OR GATE)");

OUT(E.NAVA.NAMEF);

OUT(E.NAVA.NAMET);

IWRT(E.NAVA.NOVA);

TWRT("#NL#VALUE: ");

IWRT(E.NAVA.VALUES.ACTVA);

RETURN (NILVAN);

END;

E := E.NEXT;

REP;

ENDBLOCK;

END;

END;

RETURN (CHEBC(V));

ENDPROC;

\*\*\*\*\*  
? CHECK ON-LINE VARIABLES ?

EXT PROC (RAB) INT REFATR;

ENT PROC CHEONL(REF VARIAB V) INT;

IF V.MESU = 'Y'

THEN BLOCK

REF DATAV 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.ACTVA > VI.HL OR VI.ACTVA < 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;
```

```
ENT PROC CHEVAR ( ) REF VARIAB;
  INT F := 0;
  REF VARIAB V := NILVAR;
L1: IF EST.IDEN # 'R' AND EST.IDEN # 'B'
  THEN V := FIVIS(EST.INOU);
    IF V == NILVAN
    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 == NILVAN
        THEN
          IF EST.IDEN = 'M' OR
          EST.IDEN = SP
          THEN RETURN(NILVAN);
          ELSEIF EST.NEXT.IDEN = 'M'
          OR
          EST.NEXT.IDEN = SP
          THEN
            RETURN(NILVAN);
          ELSE EST := EST.NEXT;
            V := NILVAR;
            GOTO L1;
          END;
        ELSE RETURN(V);
      END;
    END;
  END;
  RETURN (CHEBC(V));
ENDPROC;
```

2 CHECK VARIABLES BEFORE PRINTING %

EXT PROC(REF VARIAB) PRINVA;

EXT PROC(BYTE,INT) INT GETBIB;

ENT PROC CHEVBP (REF STREAM S);

REF VARIAB V := NILVAR;

REF STREAM SA := NILSTM;

IF GETBIB(S.SIO,1) = 1

THEN SA := S.FROMU.UTST;

WHILE SA :# NILSTM DO

IF SA.TOUN := S.TOUN

THEN V := SA.NAVAR;

GOTO L1;

ELSE SA := SA.NEXTS;

END;

REP;

ELSE V := S.NAVAR;

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;

ENDPROC;

\*\*\*\*\*  
X CHECK VARIABLE OUT OF RANGE X 410

```
EXT PROC ( ) REF VARIAB CHEAND;  
EXT PROC (BYTE,INT) INT GETBIB;  
EXT PROC (REF VARIAB) REF VARIAB CHEEOR;  
  
ENT PROC CHEVOR(INT F,REF VARIAB V) REF VARIAB;  
  IF F = MAXNOFAULT + 1 OR F # EST.FAULT  
    THEN IF GETBIB(GST.CLASS,1) = 1  
      THEN RETURN(CHEAND());  
      ELSE RETURN (NILVAN);  
    END;  
  END;  
  RETURN (CHEEOR(V));  
ENDPROC;
```

\* CHECK IF B. C. ARE THE SAME \*

EXT PROC (BYTE,INT) INT GETBIB;

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

ENT PROC CHSEBC(REF EVENT E) INT;

REF BOUNDECO B1 := E.BACKG.BC, B2 := BCST;

REF GATE G1 := E.BACKG, G2 := GST;

REF EVENT E1 := G1.BACKE, E2 := G2.BACKE;

INT FLAG1 := 0;

IF B1 := NILBC

THEN TWRT("#NL#NILBC IN CHSEBC.RTL");

RETURN(0);

END;

IF B1.NA1 # B2.NA1

THEN RETURN(0);

ELSEIF B1.NA2 # B2.NA2

THEN RETURN(0);

ELSEIF B1.NA3 # B2.NA3

THEN RETURN(0);

END;

WHILE E1.IDEN # 'M' DO

WHILE E2.IDEN # 'M' DO

IF GETBIB(E2.INOU,7) = 0

THEN IF E1.VAR = E2.VAR AND

E1.NAVA.NOVA = E2.NAVA.NOVA AND

E1.FAULT = E2.FAULT

THEN SBITBY(E2.INOU,7);

GOTO L1;

END;

END;

G2 := E2.BACKG;

E2 := G2.BACKE;

REP;

GOTO L2;

L1: G1 := E1.BACKG;

E1 := G1.BACKE;

G2 := GST;

E2 := G2.BACKE;

REP;

L2: E2 := GST.BACKE;

WHILE E2.IDEN # 'M' DO

IF GETBIB(E2.INOU,7) = 0

THEN FLAG1 := 1;

END;

CLBITB(E2.INOU,7);

G2 := E2.BACKG;

E2 := G2.BACKE;

REP;

IF FLAG1 = 0 AND E1.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 CLAOLI (INT NUM);

CLEGUT(NUM);

CLEVBT(NUM);

ENDPROC;

\*\*\*\*\*  
% CLEAR A BIT IN A BYTE %

ENT PROC CLBITB(REF BYTE BYT,INT BITNUM);

VAL BYT := BYTE ( BYT LAND ((255 SLL BITNUM) LOR  
(255 SRL (9-BITNUM))));

ENDPROC;

% CLEAR B. C. USED IN BUILDING TREE %

```

ENT PROC CLEBCT (INT NUM, CON);
  REF BOUNDECO B := NILEBC;
  IF NUM = 0
    THEN GOTO L1;
  END;
  FOR I := CON TO MAXNOGM1 DO
    B := BCO(I);
    B.VARBC := NILEV;
    B.NA1 := 0;
    B.NA2 := 0;
    B.NA3 := 0;
    B.NEXTBC := NILEBC;
  REP;
  FREEBC := BCO(CON);
L1: FOR I := CON TO MAXNOGM1 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;

```

ENT PROC CLEGUT (INT NUM);
  INT CON := 1;
  REF GATE G := NILGA;
  REF UNITDE N := GAT(1).BACKE.UNNO;
  IF NUM = 2
    THEN GOTO L1;
    ELSEIF NUM = 1 THEN
      WHILE N :=: NILUNI DO
        CON := CON + 1;
        N := GAT(CON).BACKE.UNNO;
      REF;
    END;
  FOR I := CON TO MAXNOGM1 DO
    G := GAT(I);
    G.CLASS := 2;
    G.EXTRA := SP;
    G.BACKE := NILEV;
    G.NEXTE := NILEV;
    G.NEGATE := NILGA;
    G.BC := NILBC;
  REP;
  FOR I := 1 TO MAXNOVAR DO
    CLBITB(VAB(I).EX,2);
  REP;
  FREEGA := GAT(CON);
  GATEFO := GAT(CON);
  L1: FOR I := CON TO MAXNOGM1 DO
    GAT(I).NEGATE := GAT(I+1);
  REP;
  CLEBCT(NUM,CON);
ENDPROC;

```

## \* CLEAR EVENTS USED IN BUILDING TREE \*

```

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;
    E.FAULT := 0;
    E.NEXT := NILEV;
    E.TOP := NILEV;
    E.BACKG := NILGA;
    E.NEXTG := NILGA;
    E.UNNO := NILUNI;
    E.PROB := 0.0BZ;
  REP;
  FREEV := EVT(CON);
  BT := 0;
  C := '+';
  FLAG := 0;
  L1: FOR I := CON TO MAXNOEM1 DO
    EVT(I).NEXT := EVT(I+1);
  REP;
ENDPROC;

```

XCLEAR A BIT IN AN INTEGER X

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

3 COMMAND INPUT 3

```
LET BUFLN = 80;
LET GMCERR = 5043;
```

```
EXT PROC (REF ARRAY BYTE) RSXGMC;
EXT PROC (INT) TESDSW;
```

```
DATA COMDAT;
  ARRAY(BUFLN) BYTE BUF := (SP(BUFLN));
  INT INEXT := 0;
  TOTAL := 0;
```

```
ENDDATA;
```

```
ENT PROC COMIP() BYTE;
  INEXT := INEXT + 1;
  IF INEXT = 1
  THEN RSXGMC(BUF);
    TESDSW(GMCERR);
    TOTAL := RSXDSW;
  END;
  IF INEXT <= TOTAL
  THEN RETURN(BUF(INEXT));
  END;
  RETURN(LF);
ENDPROC;
```

2 COMPLETE THE VARIABLE 2

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

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

GETBYT(V.NAMET,1,"NAME OF VARIABLE");

V := V.NEXVAR;

REP;

ENDBLOCK;

END;

AS := AS.NEXTS;

REP;

ENDPROC;

% COMPLETE VARIABLES FOR UNITS NOT BUILT YET %

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

EXT PROC ( ) REF VARIAB NEXTFV;

ENT PROC COMPV1(REF STREAM S);

  BYTE ANS;

  REF VARIAB V := NILVAR, VI := 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;

    VI.NEXVAR := V;

    VI := V;

    GETBYT(ANS,2,"ARE THERE ANY MORE VARIABLES(Y/N)");

  REP;

  IF ANS # 'N'

    THEN TWRT("#NL#PLEASE TYPE(Y/N)");

    GOTO L1;

  END;

ENDPROC;

\* COMPARE RABS \*

```
ENT PROC COMRAB(RAB A,B) INT;  
  IF LENGTH B > LENGTH A THEN GOTO L1;END;  
  FOR I := 1 TO LENGTH B DO  
    IF A(I) # B(I) THEN RETURN(0);END;  
  REP;  
  RETURN(1);  
L1: TWRT("#NL#STRING TOO LONG");  
  RETURN(0);  
ENDPROC;
```

× COPY BITS OF A BYTE ×

```
ENT PROC COPBIB (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;

ENT 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;

2 COPY BITS OF AN INTEGER 2

```
ENT 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(EST.NEXTG.CLASS, G.CLASS);  
  G.BACKE := "EAST";  
  EAST.NEXTG := G;  
  COPYBC(G);  
  GST := G;  
ENDPROC;
```

2 GET THE DATA OF THE VARIABLE 2

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

```
ENT PROC DATVAR(REF UNITDE U,REF STREAM S);
  REF VARIAB V := NILVAR;
  IF S.NAVAR :=: NILVAR
    THEN S.NAVAR := NEXTFV();
    SBIBY(S.EIO,1);
  END;
  V := S.NAVAR;
  WHILE GETBIB(V.EX,1) # 0 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,1,32767,1,"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 TWRT("#NL#PLEASE TYPE (Y/N)");
    GOTO L2;
  END;
  SBIBY(V.EX,1);
  IF V.NEXVAR.NAMET = SF AND V.NEXVAR.NOVA = 0
    THEN V.NEXVAR := NILVAR;
  END;
ENDPROC;
```

```
OPTION(1) BC,TR;
TITLE
DEBUG;
```

```
LET COMERR = 5200;
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) RSXOP1,RSXOP0;
EXT PROC(INT) INT RSXSW1,RSXSW0;
EXT PROC(REF ARRAY BYTE,REF ARRAY BYTE) INT TREAD;
EXT PROC() BYTE COMIP;
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,ERRLUN;
  INT RSXDSW;
ENDDATA;
```

```
DATA FILDAT;
  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 IN() # 'D' OR IN() # 'E' OR IN() # 'B' OR IN() # ' '
    THEN ERP(COMERR);
  END;
```

```
TREAD(OFILE,"");
TREAD(IFILE,"*");
I := IREAD();
J := IREAD();
RSXINT();
RSXOP0(1,OFILE);
RSXSW0(1);
RSXOP1(2,IFILE);
RSXSW1(2);
DEBUG(1,J);
RSXCL(1);
```

RSXCL(2);  
ENDPROC;

% DEBUG ARRAY OF BOUNDARY CONDITIONS %

EXT PROC(INT,INT,BYTE) PRICHA;  
EXT PROC( INT) DEBBCP;

```

ENT PROC DEBBC ( INT J);
  INT I :=1;
  TWRT("#NL(3)#ARRAY OF BOUND. COND.");
  PRICHA(1,21,'*');
  SWITCH J OF L1,L2,L3;
  TWRT("#NL#ILLÉGAL 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;
    DEBBCP(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
    L := L+1;
    UBC := BCO(L).VARBC.UNNO;
  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(1);
  TWRT("#NL(3)#B. C. : ");
  IWRT(I);
  TWRT("#NL#VARBC : ");
  IF BC.VARBC :=: 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 B.C.");
  ELSEIF BC.NEXTBC :=: NILBC
    THEN TWRT("NIL B. C.");
  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 (RAH,INT,REF BYTE) TESTSP;

EXT PROC (INT,INT,BYTE) PRICHA;

ENT 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#PRIO : ");

    K := D.PRIO;

    IWRT(K);

    TESTSP("PREX : ",1,D.PREX);

    ENDBLOCK;

  REP;

ENDPROC;

# ``` % DEBUG ARRAY OF EVENTS % ```

```

EXT PROC(INT,INT,BYTE) PRICHA;
EXT PROC( INT) DEBEVP;

```

```

ENT PROC DEBEV ( INT J);
  TWRT("#NL(3)#ARRAY OF EVENTS");
  PRICHA(1,15,'*');
  SWITCH J OF L1,L2,L3;
  TWRT("#NL#ILLÉGAL 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
      DEBEVP(I);

```

```

  REP;
  GOTO EX;

```

```

L2: BLOCK
      REF UNITDE U := EVT(1).UNNO;
      INT K := 1;
      WHILE U :=: NILUNI DO
          DEBEVP(K);
          K := K+1;
          U := EVT(K).UNNO;

```

```

      REP;
      ENDBLOCK;
      GOTO EX;

```

```

L3: BLOCK
      REF UNITDE U := EVT(1).UNNO;
      INT L := 1;
      WHILE U :=: NILUNI DO
          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 (RAB,INT,REF BYTE) TESTSP;
EXT PROC( INT) TRNOWR;

ENT PROC DEBEVP ( INT I);
  INT KU ,KI;
  REF EVENT E := EVT(1);
  TWRT("#NL(3)#EVENT : ");
  IWRT(1);
  TESTSP("IDEN: ",1,E.IDEN);
  TWRT("#NL#UNID : ");
  IF E.UNID = SP
    THEN OUT('S');
    ELSE KU := E.UNID;
      IWRT(KU);
      TWRT("#SP(3)#");
      TWRT(TYU(KU).L);
  END;
  TESTSP("VAR: ",1,E.VAR);
  TWRT("#NL#INOU: ");
  FOR I := 1 TO 8 DO
    IF GETBIB(E.INOU,I) = 1
      THEN IWRT(1);
      ELSE OUT('0');
  END;
  REP;
  TWRT("#NL#NAVA: ");
  IF E.NAVA :=: NILVAR
    THEN TWRT("NIL VARIABLE");
    ELSE IWRT(E.NAVA.NOVA);
  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 IWRT(K);
          GOTO L1;
      END;
      REP;
  END;
  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 IWRT(L);
        GOTO L2;
    END;
    REP;
  END;
  L2: TWRT("#NL#BACKG: ");
  IF E.BACKG :=: NILGA
    THEN TWRT("NIL GATE");

```

```

ELSE FOR M := 1 TO MAXNOG DO
  IF E.BACKG :=: GAT(M)
    THEN IWRT(M);
    GOTO L3;
  END;
  REP;

```

```

END;

```

```

L3: TWRT("#NL#NEXTG: ");
  IF E.NEXTG :=: NILGA
    THEN TWRT("NIL GATE");
    ELSEIF E.NEXTG :=: QM
      THEN TWRT("QM");
      ELSE FOR N := 1 TO MAXNOG DO
        IF E.NEXTG :=: GAT(N)
          THEN IWRT(N);
          GOTO L4;
        END;
        REP;
      END;

```

```

END;

```

```

L4: TWRT("#NL#UNN2: ");
  IF E.UNNO :=: NILUNI
    THEN TWRT("NIL UNIT");
    ELSE FOR P := 1 TO MAXNOUNIT DO
      IF E.UNNO :=: UNI(P)
        THEN IWRT(P);
        GOTO L5;
      END;
      REP;
    END;

```

```

END;

```

```

L5: TWRT("#NL#PROB : ");
  FWRTF(E.PROB, 4);
ENDPROC;

```

\* DEBUG & PRINT ARRAY OF FAULTS \*

EXT PROC(INT,INT,BYTE) PRICHA;

ENT PROC DEBFAU ();  
TWRT("#NL(3)#ARRAY OF FAULTS");  
PRICHA(1,15,'\*');  
FOR I := 1 TO MAXNOFAULT DO  
TWRT("#NL(2)#FAULT (");  
IWRT(I);  
TWRT(") : ");  
TWRT(FAU(I).L);

REP;

ENDPROC;

2 DEBUG ARRAY OF GATES 2

EXT PROC(INT,INT,BYTE) PRICHA;

EXT PROC( INT) DEBGAP;

ENT PROC DEBGA ( INT J);

  TWRT("#NL(3)#ARRAY OF GATES");

  PRICHA(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).BACKE.UNNO;

  INT K := 1;

  WHILE UG :=: NILUNI DO

    DEBGAP(K);

    K := K+1;

    UG := GAT(K).BACKE.UNNO;

  REP;

  ENDBLOCK;

  GOTO EX;

L3: BLOCK

  REF UNITDE UG := GAT(1).BACKE.UNNO;

  INT L := 1;

  WHILE UG :=: NILUNI DO

    L := L+1;

    UG := GAT(L).BACKE.UNNO;

  REP;

  WHILE UG :=: NILUNI DO

    DEBGAP(L);

    L := L+1;

    UG := GAT(L).BACKE.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 DEBGAP(INT I);
  INT BN;
  REF GATE GA := GAT(I);
  TWRT("#NL(3)#GATE : ");
  IWRT(I);
  TWRT("#NL#CLASS: ");
  IF GA.BACKE :=: 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: ",1,GA.EXTRA);
  TWRT("#NL#BACKE: ");
  IF GA.BACKE :=: NILEV
    THEN TWRT("NIL EVENT");
    ELSE FOR J := 1 TO MAXNOE DO
      IF GA.BACKE :=: EVT(J)
        THEN IWRT(J);
        GOTO L1;
      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;
    REP;
  END;
L2: TWRT("#NL#NEGATE: ");
  IF GA.NEGATE :=: NILNEG
    THEN TWRT("NIL NEXT GATE");
    ELSEIF GA.NEGATE :=: NILGA
      THEN TWRT("NIL GATE");
      ELSE FOR L := 1 TO MAXNOG DO
        IF GA.NEGATE :=: GAT(L)
          THEN IWRT(L);
          GOTO L3;
        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;
    REP;
  END;
  END;

```

L4: RETURN;  
ENDPROC;



2 DEBUG & PRINT ARRAY OF INDEX TO LOCATE TOP EVENTS 2

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

EXT PROC (INT,INT,BYTE) PRICHA;

ENT PROC DEBLOC();

INT TA;

TWRT("#NL(3)#ARRAY TO LOCATE TOP EVENTS");

PRICHA(1,26,'\*');

FOR I := 1 TO MAXNUNIT DO

TWRT("#NL(3)#");

TWRT("LOT: ");

IWRT(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: ",1,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 IWRT(J);

GOTO L1;

END;

REP;

END;

ENDEBLOCK;

L1:REP;

ENDPROC;

% 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#SIO: ");
      IF S.FROMU :=: NILUNI
        THEN OUT('Z');
        ELSE IWRT(GETBIB(S.SIO,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;
          REP;
      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 :=: NILSTN
        THEN TWRT("NIL NEXT STREAM");
        ELSEIF S.NEXTS :=: NILSTM
```

```
THEN TWRT("NIL STREAM");  
ELSE FOR N := 1 TO MAXNOSTR DO  
  IF S.NEXTS :=: STR(N)  
    THEN IWRT(N);  
    GOTO L4;  
END;  
REP;
```

```
END;  
ENDBLOCK;  
L4:REP;  
ENDPROC;
```

% DEBUG & PRINT ARRAY OF TYPES OF UNITS %

EXT PROC(INT,INT,BYTE) PRICHA;

ENT PROC DEBTUN ();

  TWRT("#NL(3)#ARRAY OF TYPE OF UNITS");

  PRICHA(1,23,'\*');

  FOR I := 1 TO MAXNUNIT DO

    TWRT("#NL(2)#TYPE OF UNIT (");

    IWRT(I);

    TWRT(") : ");

    TWRT(TYU(1).L);

  REP;

ENDPROC;

\* PRINT THE DATA BASE \*

EXT PROC(INT) DEBEV,DEBGA,DEBBC;

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;

TWRT("#NL#ILLÉAL 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: DEBBC(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);

DEBBC(J);

DEBLOC();

DEBUN();

DEBSTR();

DEBVAR();

DEBDAT();

DEBFAU();

DEBTUN();

Q1: RETURN;

ENDPROC;

2 DEBUG & PRINT ARRAY OF UNITS 2

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

ENT PROC DEBUN();

INT TA;

TWRT("#NL(3)#ARRAY OF UNITS");

PRICHA(1,14,'\*');

FOR I := 1 TO MAXNUNIT DO

TWRT("#NL(3)#");

TWRT("UNIT: ");

IWRT(I);

BLOCK

REF UNITDE U := UNI(I);

TWRT("#NL#UNUM: ");

IWRT(U.UNUM);

TWRT("#NL#TY: ");

IF U.TY = SP

THEN OUT('S');

ELSE TA := U.TY;

TWRT(TYU(TA).L);

END;

TESTSP("TYE: ",1,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 == NILUNI

THEN TWRT("NIL UNIT");

ELSE FOR M := 1 TO MAXNUNIT DO

IF U.NEXU == UNI(M)

THEN IWRT(M);

GOTO L3;

END;

REP;

END;  
ENDBLOCK;  
L3: REP;  
ENDPROC;

# X DEBUG & PRINT ARRAY OF VARIABLES X

```

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

```

```

ENT PROC DEBVAR();
  TWRT("#NL(3)#ARRAY OF VARIABLES");
  PRICHA(1,18,'*');
  FOR I := 1 TO MAXNOVAR DO
    TWRT("#NL(3)#");
    TWRT("VAR: ");
    IWRT(I);
  BLOCK
    REF VARIAB V := VAB(I);
    TESTSP("NAMEF: ",1,V.NAMEF);
    TESTSP("NAMET: ",1,V.NAMET);
    TESTSP("MESU : ",1,V.MESU);
    TWRT("#NL#EX :");
    IF V.NOVA = 0
      THEN OUT('Z');
      ELSE IWRT(GETBIB(V.EX,1));
    END;
    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;
      END;
    L1: TWRT("#NL#NEXVAR: ");
    IF V.NEXVAR ==: NILVAN
      THEN TWRT("NIL NEXT VAR");
      ELSEIF V.NEXVAR ==: NILVAR
        THEN TWRT("NIL VARIABLE");
        ELSE FOR K := 1 TO MAXNOVAR DO
          IF V.NEXVAR ==: VAB(K)
            THEN IWRT(K);
            GOTO L2;
          END;
          REP;
        END;
    L2: TWRT("#NL#VALUES : ");
    IF V.VALUES ==: NILDAT
      THEN TWRT("NILDAT");
      ELSE FOR L := 1 TO MAXNOMEVAR DO
        IF V.VALUES ==: DAT(L)
          THEN IWRT(L);
          GOTO L3;
        END;
        REP;
      END;
    ENDBLOCK;
    L3: REP;
  ENDPROC;

```



```
2 DELETE A B. C. 2  
ENT PROC DELBC (REF GATE G);  
  REF BOUNDECO B := G.BC;  
  B.VARBC := NILEV;  
  B.NA1 := 0;  
  B.NA2 := 0;  
  B.NA3 := 0;  
  B.NEXTBC := FREEBC;  
  FREEBC := B;  
ENDPROC;
```

% DELETE AN EVENT %

```

ENT PROC DELEV (REF EVENT A, REF GATE G) REF EVENT;
REF EVENT E := A, EI := NILEV;
IF E.UNNO := #; NILUNI OR E := #; NILEV
  THEN E.IDEN := SP;
    E.UNID := SP;
    E.VAR := SP;
    E.INOU := 0;
    E.NAVA := NILVAR;
    E.FAULT := 0;
    EI := E.NEXT;
    E.NEXT := FREEV;
    FREEV := E;
    E.TOP := NILEV;
    E.BACKG := NILGA;
    E.NEXTG := NILGA;
    UST := E.UNNO;
    E.UNNO := NILUNI;
    E.PROB := 0.0B0;
    E := EI;
  ELSE G.NEXTE := NILEV;
    E := G.BACKE;
END;
RETURN (E);
ENDPROC;

```

X DELETE A GATE X

EXT PROC (REF GATE) DELBC;

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

REF GATE G1 := G;

G := E.BACKG;

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;

```
OPTION(1) BC,TR;
TITLE
DESCRIPTION OF THE SYSTEM;
```

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

```
EXT PROC() DESUN;
EXT PROC() RSXINT;
EXT PROC(INT) RSXCL;
EXT PROC(INT,REF ARRAY BYTE) RSXOP1,RSXOP0;
EXT PROC(INT) INT RSXSW1,RSXSW0;
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(70) BYTE FILE := "#SP(70)#";
ENDDATA;
```

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

```
TREAD(FILE,"");
RSXOP0(1,FILE);
RSXSW0(1);
```

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

```
TREAD(FILE,"#LF#");
IN := ECHOIN;
RSXOP1(2,FILE);
RSXSW1(2);
DESUN ();
RSXCL(1);
RSXCL(2);
```

ENDPROC;

# X BUILDING THE TREE FOR DESIGN X

```

EXT PROC (REF INT,INT,INT,INT,RAB,LABEL) GETINT;
EXT PROC (REF BYTE,INT,RAB) GETBYT;
EXT PROC (RAB,BYTE,INT,RAB,LABEL) GETSTR;
EXT PROC (RAB,RAB) INT COMRAB;
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,'X',2,"NAME OF VARIABLE(ADD X AT THE END)",L1);
  L2:GETINT(NOVAR,1,MAXNOVAR,1,"NO. OF THE VARIABLE",L2);
  NAME := ST(1);
  NAMT := ST(2);
  L3:GETSTR(ST,'X',1,"FAULT",L3);
  FOR I := 1 TO MAXNOFAULT DO
    IF COMRAB(FAU(I).L,ST) = 1
      THEN FAULTD := 1;
      LOOKFV(FAULTD,NOVAR,NAMF,NAMT);
      RETURN;
  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 LOCAUN;
EXT PROC(REF BYTE,INT) SBITBY;
EXT PROC(REF UNITDE,REF STREAM) DESVAR;
EXT PROC(BYTE,INT) INT GETBIB;
EXT PROC(REF STREAM) REF STREAM SETSTR;

ENT 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 # '+' DO
    IF IOS = '1'
      THEN L1:GETINT(INU,1,MAXNUNIT,1,"FROM UNIT :",L1);
        IF U.INST :=: NILSTM
          THEN S := NEXTFS();
            U.INST := S;
          ELSE S := U.INST;
            IF GETBIB(S.EIO,2) # 1
              THEN S := SETSTR(S);
            ELSEIF GETBIB(S.EIO,2) = 1
              AND
              S.FROMU.UNUM # INU
              THEN S := SETSTR(S);
            END;
          END;
        SBITBY(S.SIO,1);
        S.TOUN := U;
        S.FROMU := LOCAUN(INU);
        DESVAR(U,S);
      ELSEIF IOS = '0'
        THEN L2:GETINT(OUTU,1,MAXNUNIT,1,"TO UNIT :",L2);
          IF U.OUTST :=: NILSTM
            THEN S := NEXTFS();
              U.OUTST := S;
            ELSE S := U.OUTST;
              IF GETBIB(S.EIO,2) # 1
                THEN S := SETSTR(S);
              ELSEIF GETBIB(S.EIO,2) = 1
                AND
                S.TOUN.UNUM # OUTU
                THEN S := SETSTR(S);
              END;
            END;
          SBITBY(S.SIO,2);
          S.FROMU := U;
          S.TOUN := LOCAUN(OUTU);
          DESVAR(U,S);
        ELSEIF IOS = '2'
          THEN IF U.INST :=: NILSTM
              THEN U.INST := NILINS;
            ELSEIF U.OUTST :=: NILSTM
              THEN U.OUTST :=: NILOUT;
            ELSE
              TWRT("#NL# ERROR IN ("
                "DESTR.RTL)");
            END;
          END;
        END;
  END;

```

END;

END;

GETBYT(105,1,"I/O STREAM(1-IN,0-OUT,2-DUMMY,+TERMIN)");

S.NEXTS := NILSTM;

REP;

ENDPROC;



# X DESCRIPTION OF THE UNITS X

```
EXT PROC(REF UNITDE) DESTR;
EXT PROC(INT,INT,BYTE) PRCHA;
```

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

## 2 DESCRIPTION OF VARIABLES 2

```

EXT PROC(REF BYTE,INT,RAB) GETBYT;
EXT PROC (BYTE,INT) INT GETBIB;
EXT PROC(REF UNITDE,REF STREAM) DATVAR;
EXT PROC ( ) REF VARIAB NEXTFV;
EXT PROC ( ) REF STREAM NEXTFS;
EXT PROC(REF BYTE,INT) SBITEY;
EXT PROC(REF STREAM,REF UNITDE) COMPVA;
EXT PROC(REF STREAM) COMPV1;

INT PROC DESVAR (REF UNITDE U,REF STREAM S);
  BYTE ANS,V1;
  L1:GETBYT(V1,3,"VARIABLE DESCRIPTION(Y/N)");
  WHILE V1 = 'Y' DO
    IF GETBIB(S.SIO,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 V := NEXTFV();
                      REF STREAM SA :=NEXTFS();
                      SA.FROMU := S.FROMU;
                      SA.TOUN := S.TOUN;
                      SA.NAVAR := V;
                      SA.NEXTS := NILSTM;
                      SBITEY(SA.EIO,2);
                      S.FROMU.OUTST := SA;
                      V.NEXVAR := NILVAR;
                      COMPVA(SA,U);
                      COMPV1(SA);
                    ENDELOCK;
                    ELSE COMPVA(S.FROMU.OUTST,U);
                  END;
            ELSE
              TWRT("#NL#ERROR IN INPUT(DESVAR"
                ".RTL)PLEASE TYPE(Y/N)");
              GOTO L2;
            END;
          ELSEIF GETBIB(S.SIO,1) = 0
            THEN DATVAR(U,S);
            ELSE TWRT("#NL#ERROR (DESVAR.RTL)");
          END;
        GETBYT(V1,1,"VARIABLE DESCRIPTION(Y/N)");
      REP;
      IF V1 # 'N'
        THEN TWRT("#NL(2)#PLEASE TYPE(Y/N)");
        GOTO L1;
      END;
    ENDFROC;

```

% DISPLAY MEASURED VARIABLES %

EXT PROC(INT,INT,BYTE) PRICHA;

EXT PROC(INT) PRINTN;

ENT PROC DISPLA() INT;

REF VARIAB V1 := NILVAR;

REF DATAV D1 := NILDAT;

BYTE FLAG := '0';

INT K;

PRICHA(3,26,SP);

TWRT("MEASURED VARIABLES");

PRICHA(1,26,SP);

PRICHA(0,18,'\*');

PRICHA(3,8,SP);

TWRT("VARIABLES#SP(3)#HI LIMIT#SP(3)#LO LIMIT#SP(3)#"  
"STATUS#SP(3)#PRIORITY");

PRICHA(1,8,SP);

OUT('-');

FOR I := 1 TO 3 DO

PRICHA(0,8,'-');

PRICHA(0,3,SP);

REP;

PRICHA(0,6,'-');

PRICHA(0,3,SP);

PRICHA(0,8,'-');

TWRT("#NL#");

FOR I := 1 TO MAXNOVAR DO

V1 := VAB(I);

IF V1.VALUES :=: NILDAT AND V1.MESU = 'Y'

THEN D1 := V1.VALUES;

PRICHA(1,11,SP);

OUT(V1.NAMEF);

OUT(V1.NAMET);

PRINTN(V1.NOVA);

PRICHA(0,8,SP);

PRINTN(D1.HL);

PRICHA(0,8,SP);

PRINTN(D1.LL);

PRICHA(0,7,SP);

IF D1.ACTVA > D1.HL

THEN TWRT("HI");

ELSEIF D1.ACTVA < D1.LL

THEN TWRT("LO");

ELSE TWRT("OK");

END;

PRICHA(0,8,SP);

K := D1.PRIO;

PRINTN(K);

FLAG := '1';

END;

REP;

IF FLAG = '0'

THEN PRICHA(2,10,SP);

TWRT("\*\*\* THERE ARE NO MEASURED VARIABLES IN THE"  
" SYSTEM");

RETURN(0);

END;

RETURN(1);

ENDPROC;

% 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;

2 FIND THE VARIABLE IN THE SYSTEM 2

EXT PROC (BYTE,INT) INT GETBIB;

ENT PROC FIVIS(REF BYTE IO) REF VARIAB;

REF STREAM SAU := NILSTM;

REF STREAM SIO := NILSTM;

REF VARIAB VIO := NILVAR;

L1: IF GETBIB(10,3) = 1 OR GETBIB(10,1) = 1

THEN SIO := UST:INST;

IF SIO.NEXTS := NILSTM AND

GETBIB(10,1) = 1

THEN SAU := SIO.NEXTS;

END;

ELSEIF GETBIB(10,1) = 0

THEN SIO := UST.OUTST;

END;

WHILE SIO := NILSTM DO

IF SIO.NAVAR := NILVAR OR GETBIB(10,1) = 1

THEN SIO := SIO.FROMU.OUTST;

WHILE SIO := NILSTM DO

IF SIO.TOUN := UST

THEN SIO := SIO.NEXTS

ELSE GOTO L2;

END;

REP;

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;

VIO := VIO.NEXVAR;

REP;

SIO := SIO.NEXTS;

REP;

SIO := SAU;

IF SAU := NILSTM

THEN SAU := SAU.NEXTS;

END;

REP;

TWRT("#NL#UNDEFINED VARIABLE IN THE SYSTEM (FIVIS.RTL)");

OUT(EST.VAR);

EST := EST.NEXT;

IF EST.IDEN = 'M' OR EST.IDEN = SP

THEN RETURN(NILVAN);

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,SPS;

ENT PROC NCLS (INT N);

FOR J := 1 TO N DO

TWRT("#NC#");

REP;

RETURN;

ENDPROC;

## Z GARBAGE COLLECTOR Z

```
EXT PROC (REF EVENT, REF GATE) REF EVENT DELEV;
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.NEXTE :=: NILEV
      THEN IF G.NEXTE.UNNO :=: NILUNI
        THEN E := G.NEXTE;
        END;
        WHILE E.NEXTG :=: NILGA AND
          E.UNNO :=: NILUNI DO
          E := DELEV(E, G);
        REP;
        IF E :=: NILEV
          THEN IF E.UNNO :=: NILUNI THEN GOTO L1; END;
          G := E.NEXTG;
          IF G :=: NILGA THEN GOTO L2; END;
        END;
      END;
    L1: E := G.BACKE;
    G := DELGAT(E, G);
    E := DELEV(E, G);
    EST := E;
  L2: REP;
    E1 := GST.NEXTE;
    IF E1.UNNO :=: NILUNI
      THEN WHILE E1.NEXT.UNNO :=: NILUNI DO
        E1 := E1.NEXT;
        REP;
        EAST := E1;
        EAST.NEXT := EST;
      ELSE EAST := GST.BACKE;
        GST.NEXTE := NILEV;
        C := '+';
      END;
  END;
ENDPROC;
```

2 GET A BIT IN A BYTE 2

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



Z GET A BIT IN AN INTEGER Z

```
ENT PROC GETBIT(INT WORD,BITNUM) INT;  
  RETURN((WORD SLL (16-BITNUM)) SRL 15);  
ENDPROC;
```

2 GET BYTE 2

EXT PROC(RAB,INT) OUTP;

ENT PROC GETBYT(REF BYTE B,INT LFEED,RAB AB);  
OUTP(AB,LFEED);

BLOCK

BYTE BB:= IN();

WHILE BB = CR OR BB = LF DO

BB := IN();

REP;

VAL B := BB;

ENDBLOCK;

ENDPROC;

3 GET FRACTION 3

EXT PROC (REF ARRAY BYTE,INT) OUTP;

EXT PROC ()FRAC FREAD;

EXT PROC (FRAC,FRAC,LABEL) TESTF;

ENT PROC GETFRA(REF FRAC F,FRAC LO,INT LFEED,RAB AB,LABEL ERRLAB);

FRAC A := 0.000;

OUTP(AB,LFEED);

A := FREAD();

TESTF(A,LO,ERRLAB);

VAL F := A;

ENDPROC;

\*\*\*\*\*  
X GET AN INTEGER X

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

ENT 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;

```

Z GET STRING Z

```

```

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;

REP;

TWRT("#NL#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 UNITDE) LU;  
EXT PROC(REF ARRAY VARIAB) LV;
```

```
ENT PROC LINTOP ();  
  LS(STR);  
  LU(UNI);  
  LV(VAB);  
ENDPROC;
```

```
OPTION(1) BC,TR;
TITLE
LINK TOPOLOGY;
```

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

```
EXT PROC() LINTOP;
EXT PROC() RSXINT;
EXT PROC(INT) RSXCL;
EXT PROC(INT, REF ARRAY BYTE) RSXOP1, RSXOP0;
EXT PROC(INT) INT RSXSW1, RSXSW0;
EXT PROC (REF ARRAY BYTE, REF ARRAY BYTE) INT TREAD;
EXT PROC ( ) BYTE 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, ERRUN;
  INT RSXDSW;
ENDDATA;
```

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

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

```
TREAD(OFILE, "=");
TREAD(IFILE, "#LF#");
RSXINT();
RSXOP0(1, OFILE);
RSXSW0(1);
RSXOP1(2, IFILE);
RSXSW1(2);
LINTOP();
RSXCL(1);
RSXCL(2);
ENDPROC;
```



**\* LOCATE UNITS \***

**ENT PROC LOCAUN (INT I0) REF UNITDES**

**FOR I := 1 TO CUNIT DO**

**IF UN1(I).UNUM = I0**

**THEN RETURN(UN1(I));**

**END;**

**REP;**

**TWRT("#NL#NO SUCH UNIT IN THE SYSTEM (LOCAUN.RTL)");**

**RETURN(NILUNI);**

**ENDPROC;**

3 LOOKING FOR THE VARIABLE 2

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;

ENDS

END;

REP;

TWRT("#NL#NO SUCH VARIABLE IN THE SYSTEM(LOOKFV.RTL)");

ENDPROC;

\* LOCATE THE RIGHT MINITREE \*

```
EXT PROC (BYTE,INT) INT GETBIB;
EXT PROC ( ) LFEMT,BUILD;
EXT PROC (BYTE,BYTE,INT) INT LRVMT;
```

```
ENT PROC LORIMI (INT FAULTD);
  IF BT = 0 THEN GOTO L;END;
  IF EST.IDEN = 'R'
    THEN LFEMT();
      IF LRVMT(2,SP,FAULTD) = 1
        THEN GOTO L2;
      ELSE TWRT("#NL#NO SUCH R EVENT IN UNIT NO. ");
        IWRT(UST.UNUM);
        TWRT(" LORIMI.RTL");
        TWRT("#CR#");
        RETURN;
```

END;

END;

L: IF VST := NILVAR THEN GOTO L1; END;

SST := VST.BACKS;

UST := SST.FROMU;

IF VST.NAMEF # SP AND VST.NAMET = SP AND SST.SIO = 1

THEN UST := SST.TOUN;

LFEMT();

IF LRVMT(3,VST.NAMEF,FAULTD) = 1

THEN GOTO L2;

END;

L1: TWRT("#NL#ERROR INTERNAL VAR. LORIMI.RTL#CR#");

RETURN;

END;

LFEMT();

IF LRVMT(0,VST.NAMEF,FAULTD) # 1

THEN UST := SST.TOUN;

LFEMT();

IF LRVMT(1,VST.NAMET,FAULTD) # 1

THEN TWRT("#NL#ERROR (LORIMI.RTL)");

RETURN;

END;

END;

L2: BUILD();

ENDPROC;

\* COMPARE LENGTH & COPY RABS \*

```
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);  
  REP;  
  FOR J := LENGTH B + 1 TO LENGTH A DO  
    A(J) := SP;  
  REP;  
  RETURN;  
L1: TWRT("#NL#STRING TOO LONG");  
ENDPROC;
```

\* LOCATE THE RIGHT VARIABLE IN THE MINITREE \*

EXT PROC (BYTE,INT) INT GETBIB;

```

ENT PROC LRUMT (BYTE IO,NAM,INT FAULTD) INT;
  WHILE EST :#: NILEV DO
    IF EST.VAR = NAM AND EST.FAULT = FAULTD
      THEN IF IO # 0
        THEN IF GETBIB(EST.INOU,IO) = 1
          THEN RETURN(1);
        END;
      ELSEIF GETBIB(EST.INOU,1) = 0
        THEN RETURN(1);
      END;
    END;
  END;
  EST := EST.TOP;
  REP;
  RETURN (0);
ENDPROC;

```

2 LINK STREAMS 2

```
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 MAXNUNITMI DO  
    U(I).NEXU := U(I+1);  
  REP;  
ENDPROC;
```

2 LINK VARIABLES 2

```
ENT PROC LV(REF ARRAY VARIAB V);  
  FOR I := 1 TO MAXNOVARI DO  
    V(I).NEXVAR := V(I+1);  
  REP;  
ENDPROC;
```



% DATA OF A 'M' EVENT %

EXT PROC(REF EVENT) SGA;

ENT PROC MEVT(REF EVENT E);  
SGA(E);  
ENDPROC;

\*\*\*\*\*  
 % NEW EVENT %

```

EXT PROC () REF EVENT NEXTFE;
EXT PROC (INT) LORIMI;
EXT PROC (REF BYTE,INT) SBITBY;

ENT PROC NEW EVT();
  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;
         SBITBY(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.NEXTE :=: NILEV
    THEN GST.NEXTE := E;
  END;
  IF EST.NEXTG :=: QM
    THEN LORIMI(E.FAULT);
  END;
ENDPROC;
```

% GET THE NEXT BOUNDARY CONDITION %

ENT PROC NEXTFB ( ) REF BOUNDECO;

IF FREEBC := NILNEBC THEN TWRT("#NL#NO MORE B. C. AVAILABLE");

RETURN(NILNEBC);

END;

BLOCK

REF BOUNDECO 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;
```

2 GET THE NEXT GATE 2

```
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.NEGATE;  
    RETURN(I);  
  ENDBLOCK;  
ENDPROC;
```

% GET NEXT FREE STREAM %

```
ENT PROC NEXTFS() REF STREAM;  
  IF FREEST == NILSTN  
  THEN TWRT("#NL# NO MORE STREAMS AVAILABLE IN ARRAY ");  
  END;  
  BLOCK  
    REF STREAM S := FREEST;  
    FREEST := S.NEXTS;  
    RETURN(S);  
  ENDBLOCK;  
ENDPROC;
```

\*\*\*\*\*  
Z GET NEXT FREE UNIT Z

```
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.NEXU;  
  RETURN(U);  
  ENDBLOCK;  
ENDPROC;
```

\*\*\*\*\*  
X GET NEXT FREE VARIABLE X

```
ENT PROC NEXTFV() REF VARIAB;  
IF FREEVA :=: NILVAN  
THEN TWRT("#NL# NO MORE VARIABLES AVAILABLE IN ARRAY ");  
END;  
BLOCK  
  REF VARIAB V := FREEVA;  
  FREEVA := V.NEXVAR;  
  RETURN(V);  
ENDBLOCK;  
ENDPROC;
```



2 NO NEED TO DEVELOP AN EVENT AGAIN 2

```
EXT PROC () REF EVENT NEXTFE;
EXT PROC (REF BYTE,INT) SBITBY;
```

```
ENT PROC NODEVL (REF EVENT E);
  REF EVENT EI := NEXTFE();
  EI.IDEN := EST.IDEN;
  EI.UNID := EST.UNID;
  EI.VAR := EST.VAR;
  EI.INOU := EST.INOU;
  SBITBY(EI.INOU,8);
  EI.NAVA := VST;
  EI.FAULT := EST.FAULT;
  EI.BACKG := GST;
  EI.NEXTG := E.NEXTG;
  EI.UNNO := UST;
  EI.PROB := EST.PROB;
  IF C = '*'
    THEN EAST.NEXT := EI;
    C := '+';
  END;
  EAST := EI;
  IF GST.NEXTE == NILEV
    THEN GST.NEXTE := E;
  END;
  IF EST.NEXT.IDEN = 'M' OR EST.NEXT.IDEN = SP
    THEN EI.NEXT := NILEV;
    ELSE EI.NEXT := EST.NEXT;
  END;
ENDPROC;
```

\*\*\*\*\*  
: PRINT A RAB AND PROMPT :

488

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 12;

EXT PROC() READNU;  
EXT PROC() RSXINT;  
EXT PROC(INT) RSXCL;  
EXT PROC(INT, REF ARRAY BYTE) RSXOP1, RSXOP0;  
EXT PROC(INT) INT RSXSW1, RSXSW0;  
EXT PROC (REF ARRAY BYTE, REF ARRAY BYTE) INT TREAD;  
EXT PROC ( ) BYTE 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 (35) BYTE IFILE := "#SP(35)#";  
ARRAY (35) BYTE OFILE := "#SP(35)#";  
ENDDATA;

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

TREAD(OFILE, "=");  
TREAD(IFILE, "#LF#");  
RSXINT();  
RSXOP0(1, OFILE);  
RSXSW0(1);  
RSXOP1(2, IFILE);  
RSXSW1(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(2,5,'-');
  PRICHA(0,8,SP);
  PRICHA(0,4,'-');
  PRICHA(0,4,SP);
  PRICHA(0,13,'-');
  TWRT("#NL#");
ENDPROC;

```

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

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

EXT PROC()PRIUN;  
 EXT PROC()RSXINT;  
 EXT PROC(INT)RSXCL;  
 EXT PROC(INT,REF ARRAY BYTE)RSXOP1,RSXOP0;  
 EXT PROC(INT)INT RSXSW1,RSXSW0;  
 EXT PROC (REF ARRAY BYTE,REF ARRAY BYTE) INT TREAD;  
 EXT PROC ( ) BYTE COMIF;

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

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

TREAD(OFILE,"");  
 TREAD(IFILE,"#LF#");  
 RSXINT();  
 RSXOP0(1,OFILE);  
 RSXSW0(1);  
 RSXOP1(2,IFILE);  
 RSXSW1(2);  
 PRIUN ( );  
 RSXCL(1);  
 RSXCL(2);  
 ENDPROC;

% PRINT ALL TREES %

EXT PROC () PRITRE;

ENT PROC PRIALL();

INT I := CI;

REF EVENT E := EVT(CI);

WHILE E.UNNO :# NILUNI DO

IF E.IDEN = 'M'

THEN EAST := E;

PRITRE();

END;

I := I + 1;

E := EVT(I);

REP;

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 PRIENA (INT B,NS,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 := 1 TO 16 DO
    IF GETBIT(B,J) = 1
      THEN PRICHA(1,53,SP);
      CB := 16*(N-1);
      TRNOWR(J+CB);
    END;
  REP;
  RETURN(NS);
ENDPROC;
```

\* PRINT CHARACTERS \*

EXT PROC(INT) NCLS;

ENT PROC PRICHA (INT LFEED, INT N, BYTE CHAR);

NCLS(LFEED);

FOR I := 1 TO N DO

OUT(CHAR);

REP;

ENDPROC;



% PRINT FAULT ACORDING TO THE BIT SET %

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

```
ENT PROC PRIFFB (INT B ,INT BINU);
  INT CB := 0;
  FOR I := 1 TO 16 DO
    IF GETBIT(B,I) = 1
      THEN PRICHA(1,6,SP);
      CB := 16 *(BINU-1);
      TRNOWR(1+CB);
  END;
  REP;
ENDPROC;
```

\* PRINT HEADS \*

EXT PROC (INT,INT,BYTE) PRICHA;

```

INT PROC PRIHED ();
PRICHA(2,12,SP);
TWRT("NAME NO.");
PRICHA(1,12,SP);
TWRT("VAR. VAR. FAULT#SP(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,SP);
PRICHA(0,4,'-');
TWRT("#NL#");
ENDPROC;

```

% PRINT MINITREES %

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

```

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

```

\* PRINT NOT ALLOWED COND. IN MINITREES \*

```
EXT PROC (BYTE,INT) INT GETBIB;
EXT PROC (INT,INT,BYTE) PRICHA;
EXT PROC (INT,INT,INT) INT PRIENA;
EXT PROC (INT) TRNOWR;
```

```
ENT PROC PRINAC ();
  REF BOUNDECO B := GST.BC;
  INT NS := 0;
  IF GETBIB(GST.CLASS,2) = 1
    THEN GOTO LI;
    ELSEIF GETBIB(GST.CLASS,1) = 1
      THEN PRICHA(2,2,SP);
      ELSE PRICHA(0,3,SP);
  END;
  LI: NS := PRIENA(B.NA1,NS,1);
  NS := PRIENA(B.NA2,NS,2);
  NS := PRIENA(B.NA3,NS,3);
ENDPROC;
```

EXT PROC(INT,INT,BYTE) PRICHA;

```
ENT PROC PRINOM();
  TWRT("#NL(2)#NOMENCLATURE :");
  PRICHA(1,14,'-');
  TWRT("#NL#");
  TWRT("#NL#B-EVENT : IS A BASIC EVENT AND DOES NOT REQUIRE"
    " FURTHER DEVELOPMENT");
  TWRT("#NL(2)#R-EVENT : IS AN EVENT THAT REQUIRES FURTHER"
    " DEVELOPMENT BUT IT ");
  PRICHA(1,10,SP);
  TWRT("IS RELATED TO A REPLACED EVENT IN THE UNIT");
  TWRT("#NL(2)#T-EVENT : IS AN EVENT THAT REQUIRES FURTHER"
    " DEVELOPMENT");
  TWRT("#NL#");
  PRICHA(1,67,'*');
  PRICHA(1,27,'*');
  IF RTD = 0
    THEN PRICHA(0,4,SP);
      TWRT("DESIGN");
      PRICHA(0,4,SP);
    ELSE PRICHA(0,2,SP);
      TWRT("REAL TIME");
      PRICHA(0,2,SP);
  END;
  PRICHA(0,26,'*');
  PRICHA(1,67,'*');
ENDPROC;
```

% PRINT INTEGER WITH SPACES %

ENT PROC PRINTN(INT 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.NAMEF);  
    OUT(V.NAMET);  
    IWRT(V.NOVA);  
    IF V.MESU = 'Y' THEN TWRT(" (M) ");END;  
    IF V.NEXVAR :# NILVAR THEN OUT(', ');END;  
    V := V.NEXVAR;  
  REP;  
ENDPROC;
```

% PRINT STREAMS %

EXT PROC(REF STREAM) CHEVBP;

```

ENT PROC PRIST (REF UNITDE U, BYTE IO);
  REF STREAM SU := NILSTM;
  IF IO = 'I'
    THEN SO := U.INST;
    ELSE SU := U.OUTST;
  END;
  WHILE SU :=: NILSTM DO
    IF SU :=: NILINS OR SU :=: NILOUT
      THEN TWRT("#NL#NONE");
      RETURN;
    END;
    TWRT("#NL#FROM : ");
    IWRT(SU.FROMU.UNUM);
    TWRT("#NL#TO : ");
    IWRT(SU.TOUN.UNUM);
    CHEVBP(SU);
    SU := SU.NEXTS;
  REP;
ENDPROC;

```



% PRINT THE TREE %

EXT PROC (INT,INT,BYTE) PRICHA;

EXT PROC () PRTRS,PRINOM;

ENT PROC PRITRE ();

PRINOM();

TWRT("#NL(2)#");

PRICHA(0,24,SP);

TWRT("\*\*\* FAULT TREE \*\*\*");

PRICHA(1,24,SP);

PRICHA(0,18,'-');

PRTRS();

ENDPROC;

\* PRINT TYPE OF UNIT \*

```
ENT PROC PRITYU (BYTE B);  
  INT I:= B;  
  TWRT(TYU(I).L);  
ENDPROC;
```

\* PRINT UNITS \*

```

EXT PROC(BYTE) PRITYU;
EXT PROC(REF UNITDE,BYTE) PRIST;
EXT PROC(INT,INT,BYTE) PRICHA;

ENT PROC PRIUN ();
  REF UNITDE UA := NILUNI;
  TWRT("#NL#TOPOLOGY OF THE SYSTEM");
  PRICHA(1,22,'*');
  FOR I := 1 TO COUNIT DO
    UA := UNI(I);
    PRICHA(3,65,'*');
    TWRT("#NL#UNIT NO. : ");
    IWRT(UA.UNUM);
    PRICHA(1,12,'-');
    TWRT("#NL#TYPE OF UNIT : ");
    PRITYU(UA.TY);
    TWRT("#NL##* INPUT STREAMS **");
    PRIST(UA,'1');
    TWRT("#NL##* OUTPUT STREAMS **");
    PRIST(UA,'0');
    PRICHA(1,65,'*');
  REP;
ENDPROC;

```

# ``` % PRINT VARIABLES OF MINITREES % ```

```

EXT PROC () PRINAC;
EXT PROC (BYTE,INT) INT GETBIB;
EXT PROC (INT,INT,BYTE) PRICHA;
EXT PROC (INT) TRNOWR;
EXT PROC () PRTGA;

ENT PROC PRIVAM (REF EVENT E);
  REF GATE G := NILGA;
  PRICHA(0,4,SP);
  OUT(E.VAR);
  PRICHA(0,4,SP);
  IF GETBIB(E.INOU,1) = 1
    THEN TWRT("IN");
    OUT(SF);
  ELSEIF GETBIB(E.INOU,1) = 0
    THEN TWRT("OUT");
  END;
  IF GETBIB(E.INOU,2) = 1 OR
    GETBIB(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();
      PRINAC();
      GST := G;
  END;
ENDPROC;

```

2 PRINT VARIABLE OF TREE 2

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

```

ENT PROC PRIVAT(REF EVENT E);
  IF E.NAVA :# : NILVAR
    THEN OUT(SP);
      OUT(E.NAVA.NAMEF);
      OUT(E.NAVA.NAMET);
      OUT(SP);
      OUT(SP);
      IF E.NAVA.NOVA < 10 THEN OUT(SP);END;
      IF E.NAVA.NOVA < 100 THEN OUT(SP);END;
      IWRT(E.NAVA.NOVA);
      PRICHA(0,2,SP);
    ELSE PRICHA(0,10,SP);
  END;
  TRNOWR(E.FAULT);
  TWRT("UNIT NO. ");
  IF E.UNNO.UNUM < 10 THEN OUT(SP);END;
  IWRT(E.UNNO.UNUM);
  PRICHA(0,3,SP);
  PRITYU(E.UNNO.TY);
ENDPROC;

```

% PRINT GATE OF TREE %

EXT PROC (BYTE,INT) INT GETBIB;

ENT PROC PRTGA();

OUT(SP);

IF GETBIB (GST.CLASS,2) = 1

THEN TWRT("EX-OR");

ELSEIF GETBIB(GST.CLASS,1) = 1

THEN TWRT("AND");

ELSE TWRT("OR");

END;

ENDPROC;

2 PRINT THE TREE BY STEPS 2

```

EXT PROC (REF EVENT) PRIVAT;
EXT PROC (BYTE,INT) INT GETBIB;
EXT PROC (INT,INT,BYTE) PRICHA;
EXT PROC () PRTGA,PRIHED;

ENT 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 PRICHA(0,4,SP);
      TWRT("EVENT : ");
    END;
    PRIVAT(EAST);
    IF EAST.NEXTG :# NILGA AND GETBIB(EAST.INOU,8) = 0
      THEN GST := EAST.NEXTG;
      PRTGA();
      TWRT("#NL#");
      E := GST.NEXTE;
      WHILE E :# NILEV DO
        PRICHA(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,8) = 1
        THEN PRICHA(1,32,SP);
        TWRT("*EVENT ALREADY "
          "DEVELOPED*");
        ELSE PRICHA(1,33,SP);
        TWRT("*DIAMOND EVENT*");
      END;
      IF EAST.NEXT :# NILEV
        THEN EAST := EAST.NEXT;
        GOTO L2;
      END;
    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.NEXT :# NILEV 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) RSXOP1,RSXOP0;
EXT PROC(INT) INT RSXSW1,RSXSW0;
EXT PROC (REF ARRAY BYTE,REF ARRAY BYTE) INT TREAD;
EXT PROC () BYTE 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 (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() # ' '
    THEN ERP(COMERR);
  END;

```

```

  TREAD(OFILE,"");
  TREAD(IFILE,"#LF#");
  RSXINT();
  RSXOP0(1,OFILE);
  RSXSW0(1);
  RSXOP1(2,IFILE);
  RSXSW1(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 I,INT HIORLO,LABEL ERRLAB);  
  RNGEPR(HIORLO);  
  FWRTF(I,6);  
  GOTO ERRLAB;  
ENDPROC;
```

## \* TEST RANGE OF INTEGER \*

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

```
ENT PROC RANGEPR (INT I,HIORLO,LABEL ERRLAB);  
  RNGEPR(HIORLO);  
  IWRTF(I,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;
```

```
ENT 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(1) * SP DO
    I := I + 1;
  REP;
  PRIMNT(I-1);
  FOR J := 1 TO MAXNUNIT DO
    IF LOT(J).TYPE = B
      THEN E := LOT(J).FIRSEL;
      GOTO L1;
    END;
  REP;
  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;
```

\*\*\*\*\*  
 Z READ A SET OF FAULTS Z

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

```
ENT PROC READSF ();
  INT I := COFAULT;
  L1: GETSTR(ST,'Z',1,"FAULT",L1);
  WHILE COMRAB(ST,"***") = 0 DO
    LRAB(FAU(I).L,ST);
    I := I+1;
    FOR K := 1 TO LENGTH ST DO
      ST(K) := SP;
    REP;
  L2: GETSTR(ST,'Z',1,"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,'X',1,"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,'X',1,"UNIT",L2);
  REP;
  COTYUN := I;
ENDPROC;
```

% 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, 'Z', 1, "FAULT(*0*% TO TERM. INPUT)", L1);
      IF COMRAB(ST, "*0*") = 1 THEN RETURN(0); END;
      ELSE LRAB(ST, AUX);
    END;
  FOR I := 1 TO MAXNOFAULT DO
    IF COMRAB(FAU(I).L, ST) = 1 THEN RETURN(I); END;
  REP;
  TWRT("#NL#UNDEFINED FAULT (REFATR.RTL)");
  RETURN(0);
ENDPROC;
```



% READ TYPE OF UNIT AND TRANSLATE INTO A NUMBER %

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

EXT PROC(RAB,RAB) INT COMRAB;

ENT PROC RETYUN ( ) INT;

L1:GETSTR(ST,'%',1,"TYPE OF UNIT(\*\*\*\*\*% TO TERMINATE INPUT)",L1);

IF COMRAB(ST,"\*\*\*\*\*") = 1 THEN RETURN(0);END;

FOR I := 1 TO MAXNUNIT DO

IF COMRAB(TYU(I).L,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 42;
LET LF = OCT 12;
```

```
EXT PROC() REUNIT;
EXT PROC() RSXINT;
EXT PROC(INT) RSXCL;
EXT PROC(INT, REF ARRAY BYTE) RSXOP1, RSXOP0;
EXT PROC(INT) INT RSXSW1, RSXSW0;
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, ERLUN;
  INT RSXDSW;
ENDDATA;
```

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

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

```
TREAD(FILE, "=");
RSXOP0(1, FILE);
RSXSW0(1);
```

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

```
TREAD(FILE, "#LF#");
IN := ECHOIN;
RSXOP1(2, FILE);
RSXSW1(2);
REUNIT ();
RSXCL(1);
RSXCL(2);
```

ENDPROC;

% 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;
```

```
ENT PROC REUNIT ();
  REF UNITDE U := NILUNE;
  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,1,MAXNOUNIT,1,"HOW MANY UNITS IN THE SYSTEM ",L1);
  TWRT("#NL(2)#");
  FOR I := 1 TO COUNT DO
    U := NEXTFU();
  L2: GETINT(U.UNUM,1,32767,1,"NO OF THE UNIT",L2);
    U.TY := BYTE(RETUN());
    TWRT("#NL(2)#");
  REP;
  U.NEXU := NILUNI;
ENDPROC;
```

## Z READ VALUES OF MEASURED VARIABLES Z

```

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

ENT PROC REVAME ();
  REF VARIAB V1 := NILVAR;
  REF DATAV D1 := NILDAT;
  BYTE ANS,NF,NT;
  INT N1,K;
  GETBYT(ANS,1,"DO YOU WANT TO LIST THE VARIABLES(Y/N)");
  IF ANS = 'Y' THEN K := DISPLA(); 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,'Z',2,"NAME OF VAR.(ADD Z,* TO TERM.)",LA);
  NF := ST(1);
  NT := ST(2);
  WHILE NF # '*' DO
    L1:GETINT(N1,1,MAXNOVAR,1,"NO. OF VARIABLE",L1);
    FOR I := 1 TO MAXNOVAR DO
      IF VAB(I).NOVA # N1
        THEN GOTO L2;
      ELSEIF VAB(I).NAMEF = NF AND
             VAB(I).NAMET = NT
        THEN V1 := VAB(I);
      ELSE GOTO L2;
    END;
    IF V1.VALUES :=: NILDAT AND V1.MESU = 'Y'
      THEN GOTO L3;
    ELSE TWRT("#NL#VARIABLE NOT MEASURED#CR#");
      GOTO EX;
    END;
  L2:REP;
    TWRT("#NL#NO SUCH VARIABLE IN THE SYSTEM#CR#");
    GOTO EX;
  L3:D1 := V1.VALUES;
    GETBYT(ANS,1,"DO YOU WANT TO CHANGE THE LIMITS(Y/N)");
    IF ANS = 'Y'
      THEN
        L4:GETINT(D1.HL,2,32767,1,"HIGH",L4);
        L5:GETINT(D1.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);
            D1.PRIO := BYTE(K);
          END;
        END;
    D1.PRVA := D1.ACTVA;
  L7:GETINT(D1.ACTVA,2,32767,1,"VALUE OF VAR.",L7);
  EX:GETSTR(ST,'Z',2,"NAME OF VAR.(ADD Z,* TO TERM.)",EX);
  NF := ST(1);
  NT := ST(2);

  REP;
ENDPROC;

```

2 DATA OF A 'R' EVENT 2

```
ENT PROC REVT(REF EVENT E);  
  E.BACKG := GATEPO;  
  E.NEXTG := QM;  
ENDPROC;
```

OPTION(1) BC,TR;  
 TITLE  
 READ SET OF FAULTS;

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

EXT PROC() READSF;  
 EXT PROC() RSXINT;  
 EXT PROC(INT) RSXCL;  
 EXT PROC(INT,REF ARRAY BYTE) RSXOP1,RSXOP0;  
 EXT PROC(INT) INT RSXSW1,RSXSW0;  
 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(70) BYTE FILE := "#SP(70)#";  
 ENDDATA;

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

TREAD(FILE,"");  
 RSXOP0(1,FILE);  
 RSXSW0(1);

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

TREAD(FILE,"#LF#");  
 IN := ECHOIN;  
 RSXOP1(2,FILE);  
 RSXSW1(2);  
 READSF();  
 RSXCL(1);  
 RSXCL(2);

.....  
ENDPROC;

526



\* 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() RSXINT;
EXT PROC(INT) RSXCL;
EXT PROC(INT, REF ARRAY BYTE) RSXOP1, RSXOP0;
EXT PROC(INT) INT RSXSW1, RSXSW0;
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(70) BYTE FILE := "#SF(70)#";
ENDDATA;
```

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

```
TREAD(FILE, "=");
RSXOP0(1, FILE);
RSXSW0(1);
```

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

```
TREAD(FILE, "#LF#");
IN := ECHOIN;
RSXOP1(2, FILE);
RSXSW1(2);
READTU();
RSXCL(1);
RSXCL(2);
```

.....  
ENDPROC;

```

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(INT) RSXCL;
EXT PROC(INT, REF ARRAY BYTE) RSXOP1, RSXOP0;
EXT PROC(INT) INT RSXSW1, RSXSW0;
EXT PROC (REF ARRAY BYTE, REF ARRAY BYTE) INT TREAD;
EXT PROC () BYTE 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 (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();
  RSXOP0(1, OFILE);
  RSXSW0(1);
  RSXOP1(2, IFILE);
  RSXSW1(2);
  REVAME ();
  RSXCL(1);
  RSXCL(2);
ENDPROC;

```

% SET BOUND. COND. & N.A.C. FOR MINITREE %

EXT PROC (RAB) INT REFATR;

EXT PROC(REF INT,INT) SETBIT;

ENT PROC SBC(REF BOUNDECO BOCO);

INT J;

INT FAEV := BOCO.VAREC.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;

**2 SET A BIT IN A BYTE 2**

```
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 REFATR;
EXT PROC (REF VARIAB) INT CHEVAC;
EXT PROC (BYTE,INT) INT GETBIB;
```

```
ENT PROC SCANVA();
  INT I ;
  REF VARIAB V := NILVAR, VP := NILVAR;
  WHILE 0 = 0 DO
    I := 1;
    V := VAB(I);
    WHILE V.NOVA # 0 DO
      IF V.MESU = 'Y' AND CHEVAC(V) = 1
        THEN IF GETBIB(V.EX,2) = 0
          THEN IF VP :=: NILVAR
            THEN VP := V;
          ELSEIF
            VP.VALUES.PRIO < V.VALUES.PRIO
            THEN VP := V;
        END;
      END;
    END;
    I := I+1;
    V := VAB(I);
  REP;
  IF VP :=: NILVAR THEN RETURN; END;
  VST := VP;
  BLOCK
    REF DATAV VI := VP.VALUES;
    IF VI.ACTVA > VI.HL
      THEN LORIMI(REFATR("HI"));
    ELSEIF VI.ACTVA < VI.LL
      THEN LORIMI(REFATR("LO"));
    END;
  ENDBLOCK;
  VP := NILVAR;
  BT := 0;
  C := '+';
  FLAG := 0;
  REP;
ENDPROC;
```

1 SET BOUND. COND. AND N. A. C. IN TREE 1

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

ENT PROC SETBC ();  
REF BOUNDECO B1 := EST.NEXTG.BC;  
GST.BC := NEXTFB();  
BCST := GST.BC;  
BCST.VARBC := EAST;  
COPYBI(B1.NA1, BCST.NA1);  
COPYBI(B1.NA2, BCST.NA2);  
COPYBI(B1.NA3, BCST.NA3);  
ENDPROC;



2 SET A BIT IN AN INTEGER 2

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

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

\* 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 LOR EST.NEXTG.CLASS;  
  GST.BACKE := EAST;  
  SETBC();  
ENDPROC;

% SET STREAM %

EXT PROC () REF STREAM NEXTFS;

ENT PROC SETSTR(REF STREAM S) REF STREAM;

WHILE S.NEXTS :# NILSTM DO

S := 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) GETINT;
EXT PROC () REF BOUNDECO NEXTFB;
EXT PROC () REF GATE NEXTFG;
EXT PROC (REF BOUNDECO) SBC;
EXT PROC (REF BYTE,INT) SBITBY;
```

```
ENT PROC SGA(REF EVENT E);
  INT C;
  E.NEXTG := NEXTFG();
  GATEPO := E.NEXTG;
  L1:GETINT(C,0,2,1,"TYPE OF GATE(0-OR,1-AND,2-EXOR)",L1);
  SBITBY(GATEPO.CLASS,C);
  GATEPO.BACKE := E;
  GATEPO.NEXTE := E.NEXT;
  GATEPO.BC := NEXTFB();
  GATEPO.BC.VARBC := E;
  SBC(GATEPO.BC);
ENDPROC;
```

OPTION(1) BC, TR;  
TITLE

STD. DATA;

\*\*\*\*\*

```
LET ARBSIZ = 10;
LET ARAB = ARRAY(ARBSIZ) BYTE;
LET RAB = REF ARRAY BYTE;
LET SP = OCT 40;
LET LF = OCT 12;
LET CR = OCT 15;
LET VT = OCT 13;
LET EQ = OCT 40, OCT 77, OCT 40, OCT 7, OCT 13;
LET NL = OCT 15, OCT 12;
LET FOREVER = WHILE 0=0;
```

% RESTRICTIONS OF THE DATA BASE %

\*\*\*\*\*

```
LET MAXNOE = 500;           % MAX. NO. OF EVENTS %
LET MAXNOEMI = 499;        % MAXNOE - 1 %
LET MAXNOG = 150;          % MAX. NO. OF GATES, B.C. & N.A.C. %
LET MAXNOGM1 = 149;        % MAXNOG - 1 %
LET MAXNUNIT = 15;         % MAX NO. OF UNITS %;
LET MAXNUNITM1 = 14;       % MAXNUNIT-1 %
LET MAXNOSTR = 50;         % MAX NO. OF STREAMS IN THE SYSTEM %
LET MAXNOSTRM1 = 49;       % MAXNOSTR - 1 %
LET MAXNOVAR = 50;         % MAX NO. OF VAR. IN THE SYSTEM %
LET MAXNOVARM1 = 49;       % MAXNOVAR - 1 %
LET MAXNOMEVAR = 30;       % MAX NO. OF MEASURED VARIABLES %
LET MAXNOFAULT = 50;      % MAX NO. OF FAULTS %
```

\*\*\*\*\*

% 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 BACKE, NEXTE,
            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 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 TYU ;

```

```

% NIL DECLARATIONS %

```

```

    EVENT NILEV ;

```

```

    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 COFAULT ;
INT COTYUN ;
ARRAY (ARBSIZ) BYTE ST ;

```

ENDDATA;

\*\*\*\*\*

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 NILINS ;

```

```

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 BT;
INT FAST;
INT FLAG;
REF EVENT EAST;
REF EVENT EST;
REF GATE GST;
REF BOUNDECO BCST;
REF UNITDE UST;
REF STREAM SST;
REF VARIAB VST;

```

ENDDATA;

%\*\*\*\*\*%

```

SVC DATA RRSIO;
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) IWRT;
EXT PROC() INT IREAD;
EXT PROC(RAB,RAB) INT TREAD;

```



\*\*\*\*\*  
% TEST DIRECTIVE STATUS WORD %

EXT PROC (INT) RRGL;

ENT PROC TESDSW (INT E);

IF RSXDSW < 0

THEN RRGL(E);

END;

RETURN;

ENDPROC;

2 TEST THE RANGE OF A FRACTION 2

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;

\*\*\*\*\*  
\*\*\*\*\*  
\* TEST THE RANGE OF THE INTEGER \*

EXT PROC (INT,INT,LABEL) RANGE;

ENT PROC TESTI (INT I,LO,HI,LABEL ERRLAB);

IF I < LO

THEN RANGE(LO,-1,ERRLAB);

ELSEIF I > HI

THEN RANGE(HI,+1,ERRLAB);

END;

RETURN;

ENDPROC;

\*\*\*\*\*  
\*\*\*\*\*  
% 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;
```

\*\*\*\*\*  
 .....  
 % TOP EVENT OF THE TREE %

```
EXT PROC () REF EVENT NEXTFE;
EXT PROC () SETGA;
EXT PROC (REF BYTE,INT) SBITBY;
```

```
ENT 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 := NILEV;
  SETGA();
  EAST.UNNO := UST;
  EAST.PROB := EST.PROB;
  BT := 1;
ENDPROC;
```

\* TRANSLATE NO. OF THE FAULT INTO WORDS \*

```
ENT PROC TRNOWR( INT I);  
  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) = SP THEN RETURN;  
    ELSE OUT(A(I));  
  END;  
  REP;  
ENDPROC;
```



\* GET THE VALUES OF MEASURED VARIABLES \*

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

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

ENT PROC VALUE (REF VARIAB V);

REF DATAV VA := NILDAT;

INT K;

IF COUNTV <= MAXNOMEVAR

THEN VA := DAT(COUNTV);

L1:GETINT(VA.HL,0,32767,1,"HIGH",L1);

L2:GETINT(VA.LL,0,32767,1,"LOW",L2);

L3:GETINT(K,0,255,1,"PRIORITY OF THE VAR.",L3);

VA.PRIO := BYTE(K);

V.VALUES := VA;

COUNTV := COUNTV + 1;

ELSE TWRT("#NL#NO MORE ELEMENTS IN DAT(K)"  
"AVAILABLE (VALUE.RTL)");

END;

ENDPROC;

Table III.1Look-up Table for Faults

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

Table III.2Look-up Table for Type of Units

Number Used	Type of Unit
1	CENT-PUMP
2	CLOSED-TK
3	CNTRLV-SC
4	CNTRL-VAL
5	CNTRROLLER
6	DUMMY-H
7	DUMMY-T
8	HEAT-EX
9	PIPE
10	SENSOR-Q
11	SENSOR-P
12	SENSOR-T
13	VALVE

Table III.3MODE BOUNDECO Description

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

Table III.4MODE DATAV Description

Field Name	Mode	Function
HL	INT	Stores the High limit of the variable
LL	INT	Stores the Low limit of the variable
PRVA	INT	Stores the previous value of the variable
ACTVA	INT	Stores the actual value of the variable
PRI0	BYTE	Stores the priority of the variable
PREX	BYTE	Spare field

Table III.5MODE EVENT Description

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

Table III.6MODE GATE Description

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

Table III.7MODE INDEX Description

Field Name	Mode	Function
TYPE	BYTE	Describes the type of unit.
TEX	BYTE	Spare field
FIRSEL	REF EVENT	Pointer to the first event in the set of minitrees for the unit described by the field TYPE

Table III.8MODE STREAM Description

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

Table III.9MODE UNITDE Description

Field Name	Mode	Function
UNUM	INT	Describes the No. of the unit according to the No. given in the topology of the system
TY	BYTE	Describes the type of unit
TYE	BYTE	Spare field
INST	REF STREAM	Pointer to the first input stream of the unit
OUTST	REF STREAM	Pointer to the first output stream of the unit
NEXU	REF UNITDE	Pointer to the next unit of the system

Table III.10MODE VARIAB Description

Field Name	Mode	Function
NAMEF	BYTE	Describes the first letter of the complete variable
NAMET	BYTE	Describes the other letter of the complete variable. (No letter will be assigned if it refers to an internal variable)
MESU	BYTE	Is used to know if the variable is a measured one or not
EX	BYTE	Is used only during the definition of the topology
NOVA	INT	Describes the No. of the variable
BACKS	REF STREAM	Pointer to the back stream, (the stream which carries the variable in the topology)
NEXVAR	REF VARIAB	Pointer to next variable
VALUES	REF DATVAR	Pointer to the values of the variable (only if it is a measured one)



Table III.11Description of the Pointers Defined in the Data Base

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

Table III.12Description of the Tasks

Name of the Task	Function	Options
BMT	Build Minitrees	None
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 Data Base	*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 RAB = REF ARRAY BYTE;  
LET SP = OCT 40;

% RESTRICTIONS OF THE DATA BASE %

\*\*\*\*\*

LET MAXNOE = 520;	% MAX. NO. OF EVENTS %
LET MAXNOEMI = 499;	% MAXNOE - 1 %
LET MAXNOG = 152;	% MAX. NO. OF GATES, B.C. & N.A.C. %
LET MAXNOGMI = 149;	% MAXNOG - 1 %
LET MAXNOUNIT = 15;	% MAX NO. OF UNITS %
LET MAXNOUNITMI = 14;	% MAXNOUNIT-1 %
LET MAXNOSTR = 50;	% MAX NO. OF STREAMS IN THE SYSTEM %
LET MAXNOSTRMI = 49;	% MAXNOSTR - 1 %
LET MAXNOVAR = 50;	% MAX NO. OF VAR. IN THE SYSTEM %
LET MAXNOVARMI = 49;	% MAXNOVAR - 1 %
LET MAXNOMEVAR = 30;	% MAX NO. OF MEASURED VARIABLES %
LET MAXNOFAULT = 50;	% MAX. NO. OF FAULTS %

\*\*\*\*\*

% 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, NEXTE,  
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 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);

Fig. III.2.3.1 Data Base Program: /continued

```

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

```

```

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

```

```

MODE FAS (ARAB L);

```

```

*****Z

```

```

ENT DATA MINIT;

```

```

% ARRAY OF EVENTS %

```

```

    ARRAY (MAXNOE) EVENT EVT :=
    ((SP,SP,SP,0,NILVAR,0,NILEV,NILEV,
      NILGA,NILGA,NILUNI,0.0B0) (MAXNOEM1),
      (SP,SP,SP,0,NILVAR,0,NILNEX,NILEV,
      NILGA,NILGA,NILUNI,0.0B0));

```

```

% ARRAY OF GATES %

```

```

    ARRAY (MAXNOG) GATE GAT:=
    ((0,SP,NILEV,NILEV,NILGA,NILEC) (MAXNOGM1),
      (0,SP,NILEV,NILEV,NILNEG,NILEC));

```

```

% ARRAY OF BOUND. CONDITIONS %

```

```

    ARRAY (MAXNOG) BOUNDECO BCO :=
    ((NILEV,0,0,0,NILEC) (MAXNOGM1),
      (NILEV,0,0,0,NILNEC));

```

```

% 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,
      0.0B0);

```

```

    EVENT NILNEX :=
    ('*', '*', '*', '*', NILVAN,0,NILEV,NILEV,NILNEG,NILNEG,
      NILUNI,0.0B0);

```

```

    GATE NILGA :=

```

Fig. III.2.3.1 /continued

```
(SP, SP, NILEV, NILEV, NILGA, NILBC);
```

```
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) (MAXNOUNITM1),
(0, SP, SP, NILSTM, NILSTM, NILUNE));
```

### % ARRAY OF STREAMS %

```
ARRAY (MAXNOSTR) STREAM STR :=
((0, 0, NILUNI, NILUNI, NILVAR, NILSTM) (MAXNOSTRM1),
(0, 0, NILUNI, NILUNI, NILVAR, NILSTN));
```

### % ARRAY OF VARIABLES %

```
ARRAY (MAXNOVAR) VARIAB VAB :=
((SP, SP, SP, 0, 0, NILSTM, NILVAR, NILDAT) (MAXNOVARM1),
(SP, SP, SP, 0, 0, NILSTM, NILVAN, 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, '*', '*', NILSTN, NILSTN, NILUNI);

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

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

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

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

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

VARIAB NILVAN :=
('*', '*', '*', '*', 0, NILSTN, 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 CUNIT := 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

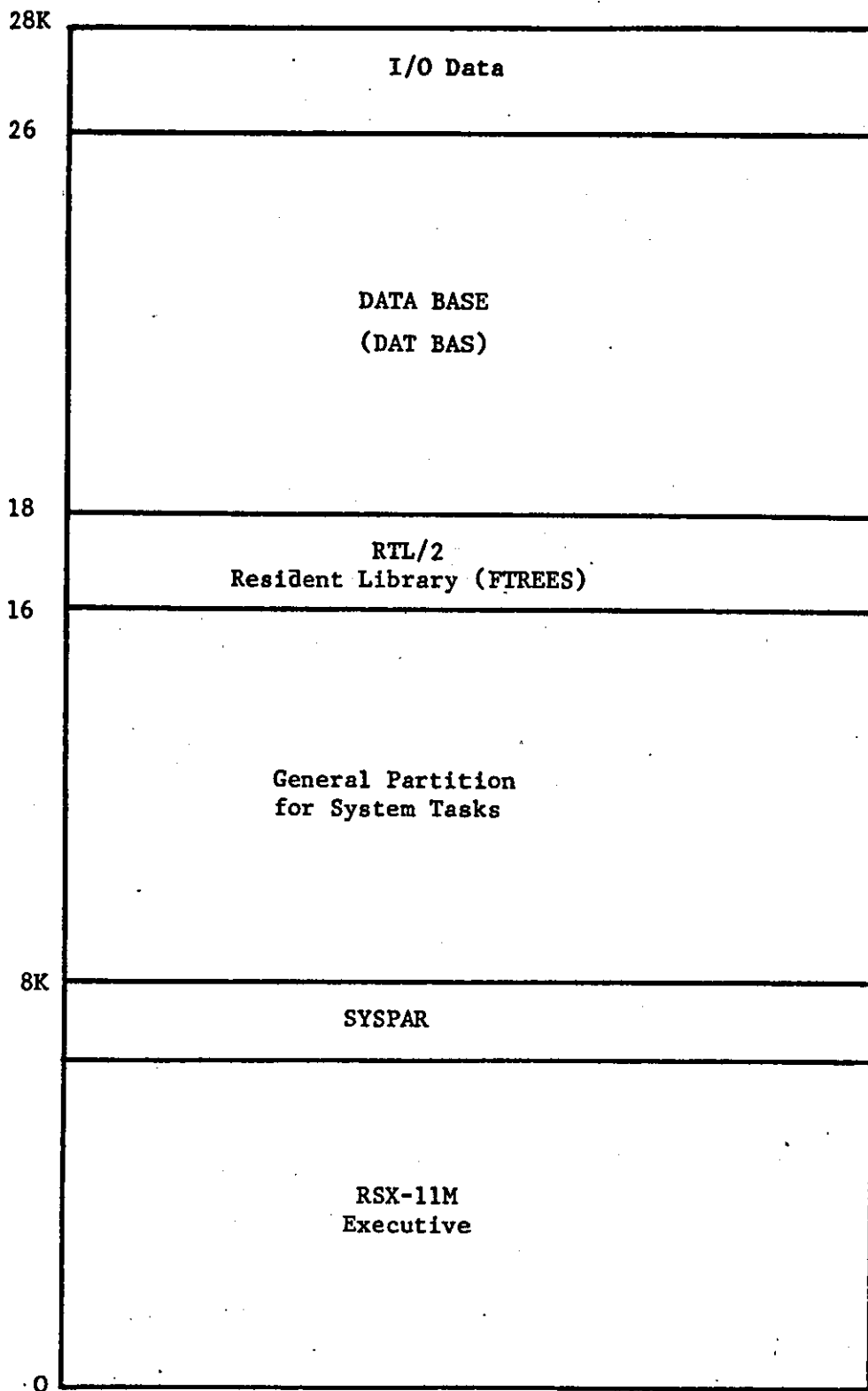


Fig. III.2.3.2 Memory layout used in this work

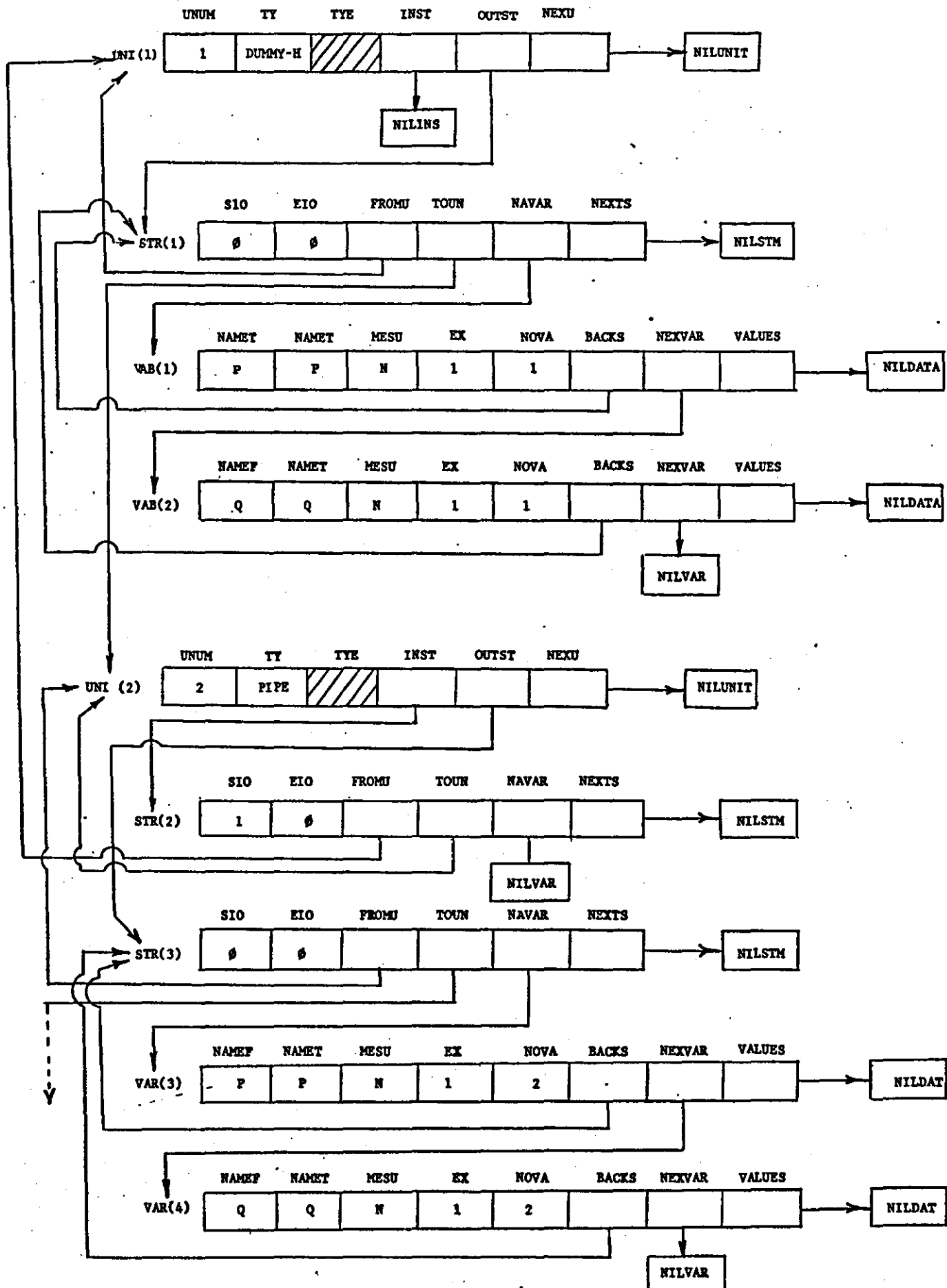


Fig. III.2.3.3 Part of the List structure for the topology of the two pipe and valve system example used in Chapter 4



BMT, TI: /SH=BA1, BMT  
IOR, AR2, T/LB, SLB/LB: SLRFR: SLRFFW, ELB/LB  
/  
UNITS=3  
ACTFIL=2  
ASG=TI:1:2:3  
PAR=GEN:40000:40000  
PRI=80  
TASK=...BMT  
COMMON=DATBAS: RW  
LIBR=FTREES: RO  
//

Fig. III.3.1 Task BMT indirect command file

BTR, TI: /SH=BTR/MP  
STACK=2220  
UNITS=3  
ACTFIL=2  
ASG=TI: 1: 2: 3  
PAR=GEN: 40000: 40000  
PRI=80  
TASK=...BTR  
COMMON=DATBAS: RW  
LIBR=FTREES: RO  
//

Fig. III.3.2 Task BTR indirect command file

```

.ROOT BTR-BA1-T/LB:ALMIP:COMIP:ALMOP:TESDSW-L1A
L1A: .FCTR ELB/LB:RSXGMC:RSXWTS:RSXQ10-L1
L1: .FCTR T/LB:BUILT:GETBIT:SETBIT:GETBIB:DESIGN:OUTP-L11
L11: .FCTR T/LB:GETSTR:COMRAB-*(L2,L3,L4)
L2: .FCTR T/LB:LOOKFV:SCANVA:LORIMI:CHEVAC-*(LA1,LA2,LA3)
LA1: .FCTR T/LB:LFEMT
LA2: .FCTR T/LB:LRVMT
LA3: .FCTR T/LB:BUILD:NEXTFE:NEXTFG:NEXTFB:DELGAT:SBITBY-L12
L12: .FCTR T/LB:DELCB:COPYBI-*(LB1,LB2)
LB1: .FCTR T/LB:TOEVT:SETGA:SETBC
LB2: .FCTR T/LB:BUILD:BUILD:NEWVT:COPYGA:COPIB-LB21
LB21: .FCTR T/LB:COPYBC:CHEVAR:CHEONL:REFATR:LRAB:FIVIS:CHEVOR-LB22
LB22: .FCTR T/LB:CHEEOR:CHEBC:CHEAND:GARCOL:DELEV:CHEEDA:NODEVL-LB23
LB23: .FCTR T/LB:CHSEBC:CLBITB:CHECKD
L3: .FCTR T/LB:GETINT:TESTI:RANGEP:RNGEPR
L4: .FCTR T/LB:GETBYT
.END

```

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:40202:40202  
PRI=80  
TASK=...CAL  
COMMON=DATEBAS:RW  
LIBR=FTREES:RO  
//

Fig. III.3.4 Task CAL indirect command file

DEB, TI: /SH=DEB/MP  
UNITS=3  
ACTFIL=2  
ASG=TI: 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

.ROOT DEB-BA1-IOR-AR2-T/LB: COMIP: TESDSW-SLB/LB: SLRFWF-L1  
L1: .FCTR ELB/LB: RSXGMC-L2  
L2: .FCTR T/LB: DEBUG: TESTSP: PRICHA-\*(L3,L4,L5,L6,L7,L8,L9,L10,L11,L12)  
L3: .FCTR T/LB: DEBEV-T/LB  
L4: .FCTR T/LB: DEBGA-T/LB  
L5: .FCTR T/LB: DEBBC-T/LB  
L6: .FCTR T/LB: DEBLOC-T/LB  
L7: .FCTR T/LB: DEBUN-T/LB  
L8: .FCTR T/LB: DEBSTR-T/LB  
L9: .FCTR T/LB: DEBVAR-T/LB  
L12: .FCTR T/LB: DEBDAT-T/LB  
L11: .FCTR T/LB: DEBFAU-T/LB  
L12: .FCTR T/LB: DEBTUN-T/LB  
.END

Fig. III.3.6 Task DEB overlay description

DES, TI: /SH=BA1, DES  
ICR, AR2, T/LB, ELB/LB  
/  
UNITS=3  
ACTFIL=2  
ASG=TI: 1: 2: 3  
PAR=GEN: 42000: 42000  
PRI=80  
TASK=...DES  
COMMON=DATBAS: RW  
LIBR=FTREES: RO  
//

Fig. III.3.7 Task DES indirect command file

LIT, TI: /SH=BA1, LIT  
IOR, AR2, T/LB, ELB/LB  
/  
UNITS=3  
ACTFIL=2  
ASG=TI: 1: 2: 3  
PAR=GEN: 40000: 40000  
PRI=80  
TASK=...LIT  
COMMON=DATEAS: RW  
LIBR=FTREES: RO  
//

Fig. III.3.8 Task LIT indirect command file



PMT, TI: /SH=BA1, PMT  
IOR, AR2, T/LB, ELB/LB  
/  
UNITS=3  
ACTFIL=2  
ASG=TI:1:2:3  
PAR=GEN:42222:42222  
PRI=82  
TASK=...PMT  
COMMON=DATBAS:RW  
LIBR=FTREES:RO  
//

Fig. III.3.9 Task PMT indirect command file

PRI, TI: /SH=BA1, PRI  
IOR, AR2, T/LB, ELB/LB  
/  
UNITS=3  
ACTFIL=2  
ASG=TI:1:2:3  
PAR=GEN:40000:40000  
PRI=80  
TASK=...PRI  
COMMON=DATBAS:RW  
LIBR=FTREES:RO  
//

Fig. III.3.10 Task PRI indirect command file

PTR, TI: /SH=BAI, PTR  
IOR, AR2, T/LB, ELB/LB  
/  
UNITS=3  
ACTFIL=2  
ASG=TI:1:2:3  
PAR=GEN:40000:40000  
PRI=80  
TASK=...PTR  
COMMON=DATBAS:RW  
LIBR=FTREES:RO  
//

Fig. III.3.11 Task PTR indirect command file

REU, T1: /SH=BA1, REU  
IOR, AR2, T/LB, ELB/LB  
/  
UNITS=3  
ACTFIL=2  
ASG=T1:1:2:3  
PAR=GEN:40000:40000  
PRI=30  
TASK=...REU  
COMMON=DATBAS:RW  
LIBR=FTREES:RO  
//

Fig. III.3.12 Task REU indirect command file

RFA, TI: /SH=BAI, RFA  
IOR, AR2, T/LB, ELB/LB  
/  
UNITS=3  
ACTFIL=2  
ASG=TI: 1: 2: 3  
PAR=GEN: 40000: 40000  
PRI=80  
TASK=...RFA  
COMMON=DATBAS: RW  
LIBR=FTREES: RO  
//

Fig. III3.13 Task RFA indirect command file

```
RTU, TI: /SH=BAI, RTU
IOR, AR2, T/LB, ELB/LB
/
UNITS=3
ACTFIL=2
ASG=TI: 1: 2: 3
PAR=GEN: 40000: 40000
PRI=80
TASK=...RTU
COMMON=DATEBAS: RW
LIBR=FTREES: RO
//
```

Fig. III 3.14 Task RTU indirect command file

```
RVA, TI: /SH=BA1, RVA  
IOR, AR2, T/LB, 2LB/LB  
/  
UNITS=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) ?  
PIPEX

TYPE OF EVENT(\* TO TERMINATE INPUT) ?

M

VARIABLE ?

Q

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

S

FAULT(\*% TO TERM. INPUT) ?

HIX

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

Z

NOT ALLOWED CONDITIONS :

FAULT(\*% TO TERM. INPUT) ?

LOCKAGEZ

FAULT(\*% TO TERM. INPUT) ?

CLOSEDZ

FAULT(\*% TO TERM. INPUT) ?

SHUTZ

FAULT(\*% TO TERM. INPUT) ?

\*%Z

TYPE OF EVENT(\* TO TERMINATE INPUT) ?

T

VARIABLE ?

P

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

I

FAULT(\*% TO TERM. INPUT) ?

HIX

TYPE OF EVENT(\* TO TERMINATE INPUT) ?

T

VARIABLE ?

P

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

S

FAULT(\*% TO TERM. INPUT) ?

LOZ

TYPE OF EVENT(\* TO TERMINATE INPUT) ?

B

VARIABLE ?

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

Z

FAULT(\*% TO TERM. INPUT) ?

FL-EX-ENVZ

PROBABILITY ?

0.0001

TYPE OF EVENT(\* TO TERMINATE INPUT) ?

M

VARIABLE ?

Q

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

2

FAULT(\*0\*% TO TERM. INPUT) ?

LO%

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

2

NOT ALLOWED CONDITIONS :

FAULT(\*0\*% TO TERM. INPUT) ?

\*0\*%

TYPE OF EVENT(\* TO TERMINATE INPUT) ?

T

VARIABLE ?

P

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

1

FAULT(\*0\*% TO TERM. INPUT) ?

LO%

TYPE OF EVENT(\* TO TERMINATE INPUT) ?

T

VARIABLE ?

P

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

2

FAULT(\*0\*% TO TERM. INPUT) ?

HI%

TYPE OF EVENT(\* TO TERMINATE INPUT) ?

B

VARIABLE ?

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

2

FAULT(\*0\*% TO TERM. INPUT) ?

BLOCKAGE%

PROBABILITY ?

0.0282

TYPE OF EVENT(\* TO TERMINATE INPUT) ?

B

VARIABLE ?

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

2

FAULT(\*0\*% TO TERM. INPUT) ?

LK-LP-ENV%

PROBABILITY ?

0.0025

TYPE OF EVENT(\* TO TERMINATE INPUT) ?

M

VARIABLE ?

P  
 2-OUT,1-IN,2-NO VAR.,3-I-VAR ?  
 1  
 FAULT(\*0\*% TO TERM. INPUT) ?  
 HIZ  
 TYPE OF GATE(2-OR,1-AND,2-EXOR) ?  
 2  
 NOT ALLOWED CONDITIONS :

FAULT(\*0\*% TO TERM. INPUT) ?  
 LK-LP-ENV%  
 FAULT(\*0\*% TO TERM. INPUT) ?  
 \*0\*%

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

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

TYPE OF EVENT(\* TO TERMINATE INPUT) ?  
 B  
 VARIABLE ?  
 2-OUT,1-IN,2-NO VAR.,3-I-VAR ?  
 2  
 FAULT(\*0\*% TO TERM. INPUT) ?  
 BLOCKAGE%  
 PROBABILITY ?  
 2.0003

TYPE OF EVENT(\* TO TERMINATE INPUT) ?  
 B  
 VARIABLE ?  
 2-OUT,1-IN,2-NO VAR.,3-I-VAR ?  
 2  
 FAULT(\*0\*% TO TERM. INPUT) ?  
 LK-HP-ENV%  
 PROBABILITY ?  
 2.0004

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

VARIABLE ?

P

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

1

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

LO%

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

2

NOT ALLOWED CONDITIONS :

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

FL-EX-ENV%

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

\*%\*%

TYPE OF EVENT(\* TO TERMINATE INPUT) ?

T

VARIABLE ?

Q

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

2

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

HI%

TYPE OF EVENT(\* TO TERMINATE INPUT) ?

T

VARIABLE ?

Q

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

1

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

LO%

TYPE OF EVENT(\* TO TERMINATE INPUT) ?

B

VARIABLE ?

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

2

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

LK-LP-ENV%

PROBABILITY ?

2.2225

TYPE OF EVENT(\* TO TERMINATE INPUT) ?

M

VARIABLE ?

T

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

2

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

HI%

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

2

NOT ALLOWED CONDITIONS :

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

\*0\*Z

TYPE OF EVENT(\* TO TERMINATE INPUT) ?

T

VARIABLE ?

T

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

1

FAULT(\*0\*Z TO TERM. INPUT) ?

HI Z

TYPE OF EVENT(\* TO TERMINATE INPUT) ?

B

VARIABLE ?

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

2

FAULT(\*0\*Z TO TERM. INPUT) ?

EXT-FIREZ

PROBABILITY ?

0.0001

TYPE OF EVENT(\* TO TERMINATE INPUT) ?

M

VARIABLE ?

T

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

0

FAULT(\*0\*Z TO TERM. INPUT) ?

LOZ

TYPE OF GATE(0-OR,1-AND,2-EXOR) ?

0

NOT ALLOWED CONDITIONS :

FAULT(\*0\*Z TO TERM. INPUT) ?

\*0\*Z

TYPE OF EVENT(\* TO TERMINATE INPUT) ?

T

VARIABLE ?

T

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

1

FAULT(\*0\*Z TO TERM. INPUT) ?

LOZ

TYPE OF EVENT(\* TO TERMINATE INPUT) ?

M

VARIABLE ?

X

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

0

FAULT(\*0\*Z TO TERM. INPUT) ?

HI Z

TYPE OF GATE(0-OR,1-AND,2-EXOR) ?

0

NOT ALLOWED CONDITIONS :

FAULT(\*S\*X TO TERM. INPUT) ?  
\*S\*X

TYPE OF EVENT(\* TO TERMINATE INPUT) ?

T  
VARIABLE ?

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

1  
FAULT(\*S\*X TO TERM. INPUT) ?  
HI X

TYPE OF EVENT(\* TO TERMINATE INPUT) ?

M  
VARIABLE ?

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

0  
FAULT(\*S\*X TO TERM. INPUT) ?  
LO X

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

0  
NOT ALLOWED CONDITIONS :

FAULT(\*S\*X TO TERM. INPUT) ?  
\*S\*X

TYPE OF EVENT(\* TO TERMINATE INPUT) ?

T  
VARIABLE ?

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

1  
FAULT(\*S\*X TO TERM. INPUT) ?  
LO X

TYPE OF EVENT(\* TO TERMINATE INPUT) ?

M  
VARIABLE ?

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

5  
FAULT(\*S\*X TO TERM. INPUT) ?  
NO-FLOW X

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

0  
NOT ALLOWED CONDITIONS :

FAULT(\*S\*X TO TERM. INPUT) ?  
\*S\*X

TYPE OF EVENT(\* TO TERMINATE INPUT) ?

T  
VARIABLE ?

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

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

0-OUT,1-IN,2-NO VAR.,3-1-VAR ?  
 2  
 FAULT(\*0\*% TO TERM. INPUT) ?  
 COMP-BLOC%  
 PROBABILITY ?  
 0.0001

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

Q  
 0-OUT,1-IN,2-NO VAR.,3-1-VAR ?  
 0  
 FAULT(\*0\*% TO TERM. INPUT) ?  
 GT0%  
 TYPE OF GATE(0-OR,1-AND,2-EXOR) ?  
 0  
 NOT ALLOWED CONDITIONS :

FAULT(\*0\*% TO TERM. INPUT) ?  
 \*0\*%

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

Q  
 0-OUT,1-IN,2-NO VAR.,3-1-VAR ?  
 1  
 FAULT(\*0\*% TO TERM. 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 ?

0

FAULT(\*0\* TO TERM. INPUT) ?

HI

TYPE OF GATE(0-OR, 1-AND, 2-EXOR) ?

0

NOT ALLOWED CONDITIONS :

FAULT(\*0\* TO TERM. INPUT) ?

\*0\*

TYPE OF EVENT(\* TO TERMINATE INPUT) ?

T

VARIABLE ?

T

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

1

FAULT(\*0\* TO TERM. INPUT) ?

HI

TYPE OF EVENT(\* TO TERMINATE INPUT) ?

T

VARIABLE ?

Q

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

0

FAULT(\*0\* TO TERM. INPUT) ?

HI

TYPE OF EVENT(\* TO TERMINATE INPUT) ?

R

VARIABLE ?

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

2

FAULT(\*0\* TO TERM. INPUT) ?

Z-LO

TYPE OF EVENT(\* TO TERMINATE INPUT) ?

B

VARIABLE ?

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

2

FAULT(\*0\* TO TERM. INPUT) ?

EXT-FIRE

PROBABILITY ?

0.0001

TYPE OF EVENT(\* TO TERMINATE INPUT) ?

M

VARIABLE ?

T

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

0

FAULT(\*0\*2 TO TERM. INPUT) ?

LOZ

TYPE OF GATE(0-OR, 1-AND, 2-EXOR) ?

0

NOT ALLOWED CONDITIONS :

FAULT(\*0\*2 TO TERM. INPUT) ?

\*0\*2

TYPE OF EVENT(\* TO TERMINATE INPUT) ?

T

VARIABLE ?

T

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

1

FAULT(\*0\*2 TO TERM. INPUT) ?

LOZ

TYPE OF EVENT(\* TO TERMINATE INPUT) ?

R

VARIABLE ?

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

2

FAULT(\*0\*2 TO TERM. INPUT) ?

AZ

TYPE OF EVENT(\* TO TERMINATE INPUT) ?

R

VARIABLE ?

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

2

FAULT(\*0\*2 TO TERM. INPUT) ?

Z-HIZ

TYPE OF EVENT(\* TO TERMINATE INPUT) ?

M

VARIABLE ?

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

2

FAULT(\*0\*2 TO TERM. INPUT) ?

AZ

TYPE OF GATE(0-OR, 1-AND, 2-EXOR) ?

2

NOT ALLOWED CONDITIONS :

FAULT(\*0\*2 TO TERM. INPUT) ?

\*8\*X

TYPE OF EVENT(\* TO TERMINATE INPUT) ?

T

VARIABLE ?

Q

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

S

FAULT(\*8\*X TO TERM. INPUT) ?

LOX

TYPE OF EVENT(\* TO TERMINATE INPUT) ?

T

VARIABLE ?

Q

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

S

FAULT(\*8\*X TO TERM. INPUT) ?

NO-FLOWX

TYPE OF EVENT(\* TO TERMINATE INPUT) ?

M

VARIABLE ?

C

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

S

FAULT(\*8\*X TO TERM. INPUT) ?

HI X

TYPE OF GATE(0-OR,1-AND,2-EXOR) ?

S

NOT ALLOWED CONDITIONS :

FAULT(\*8\*X TO TERM. INPUT) ?

\*8\*X

TYPE OF EVENT(\* TO TERMINATE INPUT) ?

T

VARIABLE ?

C

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

I

FAULT(\*8\*X TO TERM. INPUT) ?

HI X

TYPE OF EVENT(\* TO TERMINATE INPUT) ?

R

VARIABLE ?

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

2

FAULT(\*8\*X TO TERM. INPUT) ?

BX

TYPE OF EVENT(\* TO TERMINATE INPUT) ?

R

VARIABLE ?

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

2

FAULT(\*0\*% TO TERM. INPUT) ?

Z-H1Z

TYPE OF EVENT(\* TO TERMINATE INPUT) ?

B

VARIABLE ?

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

2

FAULT(\*0\*% TO TERM. INPUT) ?

EXT-FIREZ

PROBABILITY ?

0.0002

TYPE OF EVENT(\* TO TERMINATE INPUT) ?

M

VARIABLE ?

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

2

FAULT(\*0\*% TO TERM. INPUT) ?

BZ

TYPE OF GATE(0-OR,1-AND,2-EXOR) ?

2

NOT ALLOWED CONDITIONS :

FAULT(\*0\*% TO TERM. INPUT) ?

\*0\*%

TYPE OF EVENT(\* TO TERMINATE INPUT) ?

T

VARIABLE ?

B

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

0

FAULT(\*0\*% TO TERM. INPUT) ?

LOZ

TYPE OF EVENT(\* TO TERMINATE INPUT) ?

R

VARIABLE ?

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

2

FAULT(\*0\*% TO TERM. INPUT) ?

CZ

TYPE OF EVENT(\* TO TERMINATE INPUT) ?

M

VARIABLE ?

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

2  
 FAULT(\*8\*X TO TERM. INPUT) ?  
 CX  
 TYPE OF GATE(0-OR,1-AND,2-EXOR) ?  
 1  
 NOT ALLOWED CONDITIONS :

FAULT(\*8\*X TO TERM. INPUT) ?  
 \*8\*X

TYPE OF EVENT(\* TO TERMINATE INPUT) ?  
 T  
 VARIABLE ?  
 B  
 2-OUT,1-IN,2-NO VAR.,3-I-VAR ?  
 0  
 FAULT(\*8\*X TO TERM. INPUT) ?  
 NO-FLOWX

TYPE OF EVENT(\* TO TERMINATE INPUT) ?  
 T  
 VARIABLE ?  
 Q  
 0-OUT,1-IN,2-NO VAR.,3-I-VAR ?  
 0  
 FAULT(\*8\*X TO TERM. INPUT) ?  
 GT0X

TYPE OF EVENT(\* TO TERMINATE INPUT) ?  
 M  
 VARIABLE ?  
 Q  
 2-OUT,1-IN,2-NO VAR.,3-I-VAR ?  
 0  
 FAULT(\*8\*X TO TERM. INPUT) ?  
 GT0X  
 TYPE OF GATE(0-OR,1-AND,2-EXOR) ?  
 0  
 NOT ALLOWED CONDITIONS :  
 FAULT(\*8\*X TO TERM. INPUT) ?  
 \*8\*X

TYPE OF EVENT(\* TO TERMINATE INPUT) ?  
 T  
 VARIABLE ?  
 Q  
 2-OUT,1-IN,2-NO VAR.,3-I-VAR ?  
 1  
 FAULT(\*8\*X TO TERM. INPUT) ?  
 GT0X

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

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

0

FAULT(\*0\*Z TO TERM. INPUT) ?

LOZ

TYPE OF GATE(0-OR, 1-AND, 2-EXOR) ?

0

NOT ALLOWED CONDITIONS :

FAULT(\*0\*Z TO TERM. INPUT) ?

\*0\*Z

TYPE OF EVENT(\* TO TERMINATE INPUT) ?

T

VARIABLE ?

C

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

1

FAULT(\*0\*Z TO TERM. INPUT) ?

LOZ

TYPE OF EVENT(\* TO TERMINATE INPUT) ?

T

VARIABLE ?

B

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

0

FAULT(\*0\*Z TO TERM. INPUT) ?

LOZ

TYPE OF EVENT(\* TO TERMINATE INPUT) ?

R

VARIABLE ?

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

2

FAULT(\*0\*Z TO TERM. INPUT) ?

Z-LOZ

TYPE OF EVENT(\* TO TERMINATE INPUT) ?

M

VARIABLE ?

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

2

FAULT(\*0\*Z TO TERM. INPUT) ?

Z-HIZ

TYPE OF GATE(0-OR, 1-AND, 2-EXOR) ?

0

NOT ALLOWED CONDITIONS :

FAULT(\*0\*Z TO TERM. INPUT) ?

Z-LOZ

FAULT(\*0\*Z TO TERM. INPUT) ?

\*0\*Z

TYPE OF EVENT(\* TO TERMINATE INPUT) ?

T  
VARIABLE ?  
T  
2-OUT, 1-IN, 2-NO VAR., 3-I-VAR ?  
S  
FAULT(\*S\*X TO TERM. INPUT) ?  
HI X

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

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

2-OUT, 1-IN, 2-NO VAR., 3-I-VAR ?  
2  
FAULT(\*S\*X TO TERM. INPUT) ?  
Z-LO X  
TYPE OF GATE(2-OR, 1-AND, 2-EXOR) ?  
2  
NOT ALLOWED CONDITIONS :

FAULT(\*S\*X TO TERM. INPUT) ?  
Z-HI X  
FAULT(\*S\*X TO TERM. INPUT) ?  
\*S\*X

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

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

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

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

1  
FAULT(\*8\*X TO TERM. INPUT) ?

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

2  
NOT ALLOWED CONDITIONS :

FAULT(\*8\*X TO TERM. INPUT) ?

LK-LP-ENVZ

FAULT(\*8\*X TO TERM. INPUT) ?

\*8\*X

TYPE OF EVENT(\* TO TERMINATE INPUT) ?

T

VARIABLE ?

Q

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

1

FAULT(\*8\*X TO TERM. INPUT) ?

HIZ

TYPE OF EVENT(\* TO TERMINATE INPUT) ?

T

VARIABLE ?

Q

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

2

FAULT(\*8\*X TO TERM. INPUT) ?

LOZ

TYPE OF EVENT(\* TO TERMINATE INPUT) ?

B

VARIABLE ?

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

2

FAULT(\*8\*X TO TERM. INPUT) ?

LOCKAGEZ

PROBABILITY ?

2.2283

TYPE OF EVENT(\* TO TERMINATE INPUT) ?

B

VARIABLE ?

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

2

FAULT(\*8\*X TO TERM. INPUT) ?

LK-HP-ENVZ

PROBABILITY ?

2.2284

TYPE OF EVENT(\* TO TERMINATE INPUT) ?

M

VARIABLE ?



P  
 0-OUT, 1-IN, 2-NO VAR., 3-1-VAR ?  
 I  
 FAULT(\*0\*X TO TERM. INPUT) ?  
 LOZ  
 TYPE OF GATE(0-OR, 1-AND, 2-EXOR) ?  
 0  
 NOT ALLOWED CONDITIONS :

FAULT(\*0\*X TO TERM. INPUT) ?  
 FL-EX-ENVX  
 FAULT(\*0\*X TO TERM. INPUT) ?  
 \*0\*X

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

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

TYPE OF EVENT(\* TO TERMINATE INPUT) ?  
 B  
 VARIABLE ?  
 0-OUT, 1-IN, 2-NO VAR., 3-1-VAR ?  
 2  
 FAULT(\*0\*X TO TERM. INPUT) ?  
 LK-LP-ENVX  
 PROBABILITY ?  
 0.0005

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

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

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

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

TYPE OF EVENT(\* TO TERMINATE INPUT) ?  
 B  
 VARIABLE ?  
 0-OUT, 1-IN, 2-NO VAR., 3-I-VAR ?  
 2  
 FAULT(\*%\* TO TERM. INPUT) ?  
 BLOCKAGE%  
 PROBABILITY ?  
 0.0006

TYPE OF EVENT(\* TO TERMINATE INPUT) ?  
 B  
 VARIABLE ?  
 0-OUT, 1-IN, 2-NO VAR., 3-I-VAR ?  
 2  
 FAULT(\*%\* TO TERM. INPUT) ?  
 LK-HP-ENV%  
 PROBABILITY ?  
 0.0017

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

FL-EX-ENVZ

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

\*%\*%

TYPE OF EVENT(\* TO TERMINATE INPUT) ?

T

VARIABLE ?

B

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

1

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

LOZ

TYPE OF EVENT(\* TO TERMINATE INPUT) ?

T

VARIABLE ?

B

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

0

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

HI%

TYPE OF EVENT(\* TO TERMINATE INPUT) ?

B

VARIABLE ?

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

2

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

LK-LP-ENVZ

PROBABILITY ?

0.0027

TYPE OF EVENT(\* TO TERMINATE INPUT) ?

M

VARIABLE ?

Q

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

0

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

HI%

TYPE OF GATE(0-OR, 1-AND, 2-EXOR) ?

0

NOT ALLOWED CONDITIONS :

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

BLOCKAGEZ

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

CLOSEDZ

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

SHUTZ

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

\*%\*%

TYPE OF EVENT(\* TO TERMINATE INPUT) ?

T

VARIABLE ?

P

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

1

FAULT(\*0\*2 TO TERM. INPUT) ?

HI%

TYPE OF EVENT(\* TO TERMINATE INPUT) ?

T

VARIABLE ?

P

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

0

FAULT(\*0\*2 TO TERM. INPUT) ?

LO%

TYPE OF EVENT(\* TO TERMINATE INPUT) ?

B

VARIABLE ?

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

2

FAULT(\*0\*2 TO TERM. INPUT) ?

FL-EX-ENV%

PROBABILITY ?

2.0000

TYPE OF EVENT(\* TO TERMINATE INPUT) ?

M

VARIABLE ?

Q

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

0

FAULT(\*0\*2 TO TERM. INPUT) ?

LO%

TYPE OF GATE(0-OR, 1-AND, 2-EXOR) ?

0

NOT ALLOWED CONDITIONS :

FAULT(\*0\*2 TO TERM. INPUT) ?

\*0\*2

TYPE OF EVENT(\* TO TERMINATE INPUT) ?

T

VARIABLE ?

P

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

1

FAULT(\*0\*2 TO TERM. INPUT) ?

LO%

TYPE OF EVENT(\* TO TERMINATE INPUT) ?

T

VARIABLE ?

P

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

3  
 FAULT(\*8\*X TO TERM. INPUT) ?  
 HI X

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

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

2  
 FAULT(\*8\*X TO TERM. INPUT) ?  
 BLOCKAGE X  
 PROBABILITY ?  
 2.2229

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

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

2  
 FAULT(\*8\*X TO TERM. INPUT) ?  
 LK-LP-ENV X  
 PROBABILITY ?  
 2.2212

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

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

2  
 FAULT(\*8\*X TO TERM. INPUT) ?  
 HI X  
 TYPE OF GATE(2-OR, 1-AND, 2-EXOR) ?

2  
 NOT ALLOWED CONDITIONS :

FAULT(\*8\*X TO TERM. INPUT) ?  
 BLOCKAGE X  
 FAULT(\*8\*X TO TERM. INPUT) ?  
 CLOSED X  
 FAULT(\*8\*X TO TERM. INPUT) ?  
 SHUT X  
 FAULT(\*8\*X TO TERM. INPUT) ?  
 \*8\*X

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

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

1  
 FAULT(\*8\*X TO TERM. INPUT) ?  
 HI X

TYPE OF EVENT(\* TO TERMINATE INPUT) ?

T

VARIABLE ?

A

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

2

FAULT(\*0\*% TO TERM. INPUT) ?

LOZ

TYPE OF EVENT(\* TO TERMINATE INPUT) ?

B

VARIABLE ?

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

2

FAULT(\*0\*% TO TERM. INPUT) ?

FL-EX-ENVZ

PROBABILITY ?

2.0011

TYPE OF EVENT(\* TO TERMINATE INPUT) ?

M

VARIABLE ?

B

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

8

FAULT(\*0\*% TO TERM. INPUT) ?

LOZ

TYPE OF GATE(0-OR, 1-AND, 2-EXOR) ?

8

NOT ALLOWED CONDITIONS :

FAULT(\*0\*% TO TERM. INPUT) ?

\*0\*%

TYPE OF EVENT(\* TO TERMINATE INPUT) ?

T

VARIABLE ?

A

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

1

FAULT(\*0\*% TO TERM. INPUT) ?

LOZ

TYPE OF EVENT(\* TO TERMINATE INPUT) ?

T

VARIABLE ?

A

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

5

FAULT(\*0\*% TO TERM. INPUT) ?

HI Z

TYPE OF EVENT(\* TO TERMINATE INPUT) ?

B

VARIABLE ?

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

2

FAULT(\*0\*X TO TERM. INPUT) ?

BLOCKAGE%

PROBABILITY ?

0.0012

TYPE OF EVENT(\* TO TERMINATE INPUT) ?

B

VARIABLE ?

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

2

FAULT(\*0\*X TO TERM. INPUT) ?

LK-LP-ENV%

PROBABILITY ?

0.0013

TYPE OF EVENT(\* TO TERMINATE INPUT) ?

M

VARIABLE ?

X

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

0

FAULT(\*0\*X TO TERM. INPUT) ?

HIZ

TYPE OF GATE(0-OR, 1-AND, 2-EXOR) ?

0

NOT ALLOWED CONDITIONS :

FAULT(\*0\*X TO TERM. INPUT) ?

\*0\*X

TYPE OF EVENT(\* TO TERMINATE INPUT) ?

T

VARIABLE ?

X

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

1

FAULT(\*0\*X TO TERM. INPUT) ?

HIZ

TYPE OF EVENT(\* TO TERMINATE INPUT) ?

M

VARIABLE ?

X

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

0

FAULT(\*0\*X TO TERM. INPUT) ?

LOX

TYPE OF GATE(0-OR, 1-AND, 2-EXOR) ?

0

NOT ALLOWED CONDITIONS :

FAULT(\*0\*X TO TERM. INPUT) ?

\*0\*X

TYPE OF EVENT(\* TO TERMINATE INPUT) ?

T  
VARIABLE ?

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

1  
FAULT(\*0%X TO TERM. INPUT) ?  
LOZ

TYPE OF EVENT(\* TO TERMINATE INPUT) ?

M  
VARIABLE ?

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

0  
FAULT(\*0%X TO TERM. INPUT) ?  
HI Z

TYPE OF GATE(0-OR, 1-AND, 2-EXOR) ?  
0

NOT ALLOWED CCNDITIONS :

FAULT(\*0%X TO TERM. INPUT) ?  
\*0%X

TYPE OF EVENT(\* TO TERMINATE INPUT) ?

T  
VARIABLE ?

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

1  
FAULT(\*0%X TO TERM. INPUT) ?  
HI X

TYPE OF EVENT(\* TO TERMINATE INPUT) ?

M  
VARIABLE ?

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

0  
FAULT(\*0%X TO TERM. INPUT) ?  
LOZ

TYPE OF GATE(0-OR, 1-AND, 2-EXOR) ?  
0

NOT ALLOWED CONDITIONS :

FAULT(\*0%X TO TERM. INPUT) ?  
\*0%X

TYPE OF EVENT(\* TO TERMINATE INPUT) ?

T  
VARIABLE ?

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

1  
FAULT(\*0%X TO TERM. INPUT) ?



LOX

TYPE OF EVENT(\* TO TERMINATE INPUT) ?

M

VARIABLE ?

B

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

8

FAULT(\*8\*% TO TERM. INPUT) ?

NO-FLOW%

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

8

NOT ALLOWED CONDITIONS :

FAULT(\*8\*% TO TERM. INPUT) ?

\*8\*%

TYPE OF EVENT(\* TO TERMINATE INPUT) ?

T

VARIABLE ?

B

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

1

FAULT(\*8\*% TO TERM. INPUT) ?

NO-FLOW%

TYPE OF EVENT(\* TO TERMINATE INPUT) ?

B

VARIABLE ?

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

2

FAULT(\*8\*% TO TERM. INPUT) ?

COMP-BLOC%

PROBABILITY ?

2.2221

TYPE OF EVENT(\* TO TERMINATE INPUT) ?

M

VARIABLE ?

Q

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

8

FAULT(\*8\*% TO TERM. INPUT) ?

NO-FLOW%

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

8

NOT ALLOWED CONDITIONS :

FAULT(\*8\*% TO TERM. INPUT) ?

\*8\*%

TYPE OF EVENT(\* TO TERMINATE INPUT) ?

T

VARIABLE ?

Q

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

1  
 FAULT(\*2\*% TO TERM. INPUT) ?  
 NO-FLOW%

TYPE OF EVENT(\* TO TERMINATE INPUT) ?

B  
 VARIABLE ?

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

2  
 FAULT(\*2\*% TO TERM. INPUT) ?  
 COMP-BLOC%  
 PROBABILITY ?  
 2.2231

TYPE OF EVENT(\* TO TERMINATE INPUT) ?

M  
 VARIABLE ?

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

0  
 FAULT(\*2\*% TO TERM. INPUT) ?  
 GT0%  
 TYPE OF GATE(2-OR, 1-AND, 2-EXOR) ?  
 0  
 NOT ALLOWED CONDITIONS :

FAULT(\*2\*% TO TERM. INPUT) ?  
 \*2\*%

TYPE OF EVENT(\* TO TERMINATE INPUT) ?

T  
 VARIABLE ?

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

1  
 FAULT(\*2\*% TO TERM. INPUT) ?  
 GT0%

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 : 1

\*\*\*\*\*

DESCRIPTION OF THE UNITS

-----

HOW MANY UNITS IN THE SYSTEM ?

5

NO OF THE UNIT ?

1

TYPE OF UNIT(\*\*\*\*\* TO TERMINATE INPUT) ?

DUMMY-HZ

NO OF THE UNIT ?

2

TYPE OF UNIT(\*\*\*\*\* TO TERMINATE INPUT) ?

PIPEX

NO OF THE UNIT ?

3

TYPE OF UNIT(\*\*\*\*\* TO TERMINATE INPUT) ?

VALVEX

NO OF THE UNIT ?

4

TYPE OF UNIT(\*\*\*\*\* TO TERMINATE INPUT) ?

PIPEX

NO OF THE UNIT ?

5

TYPE OF UNIT(\*\*\*\*\* TO TERMINATE INPUT) ?

DUMMY-TZ

LISTING

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) ?

2

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,2-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,2-OUT,2-DUMMY,+--TERMIN) ?

+

OUTPUT STREAMS

-----  
 .....

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

+

UNIT NO. : 3 VALVE

\*\*\*\*\*

INPUT STREAMS

-----  
 .....

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

1  
FROM UNIT : ?

2

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,2-OUT,2-DUMMY,+--TERMIN) ?

2

TO UNIT : ?

4

VARIABLE 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

VARIABLE DESCRIPTION(Y/N) ?

Y

NAME OF VARIABLE ?

Q

NO. OF THE VARIABLE ?

3

IS THE VARIABLE MEASURED(Y/N) ?

N

VARIABLE DESCRIPTION(Y/N) ?

N

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

+

UNIT NO. : 4 PIPE

\*\*\*\*\*

INPUT STREAMS



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

1  
 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  
 NO. OF THE VARIABLE ?

4  
 IS THE VARIABLE MEASURED(Y/N) ?

N  
 VARIABLE DESCRIPTION(Y/N) ?

Y  
 NAME OF VARIABLE ?

Q  
 NO. OF THE VARIABLE ?

4  
 IS THE VARIABLE MEASURED(Y/N) ?

Y  
 HIGH ?

100

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

\*\*\*\*\*

## 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) ?

+

LISTING

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 :

HI

NA2 :  
 OTHER-CAU  
 NA3 :  
 NEXTBC: 7

B. C. : 7  
 VARBC : 13  
 NA1 :  
 LO  
 NA2 :  
 OTHER-CAU  
 NA3 :  
 NEXTBC: 8

B. C. : 8  
 VARBC : 15  
 NA1 :  
 HI  
 NA2 :  
 OTHER-CAU  
 NA3 :  
 NEXTBC: 9

B. C. : 9  
 VARBC : 17  
 NA1 :  
 NA2 :  
 OTHER-CAU  
 NA3 :  
 NEXTBC: 10

B. C. : 10  
 VARBC : 19  
 NA1 :  
 NA2 :  
 OTHER-CAU  
 NA3 :  
 NEXTBC: 11

B. C. : 11  
 VARBC : 21  
 NA1 :  
 LO  
 LK-LP-ENV  
 NA2 :  
 NA3 :  
 NEXTBC: 12

B. C. : 12  
 VARBC : 23  
 NA1 :  
 HI  
 FL-EX-ENV  
 NA2 :

NA3 :  
NEXTBC: 13

B. C. : 13  
VARBC : 25  
NA1 :  
    LO  
    CLOSED  
    BLOCKAGE  
    SHUT

NA2 :  
NA3 :  
NEXTBC: 14

B. C. : 14  
VARBC : 29  
NA1 :  
    HI

NA2 :  
NA3 :  
NEXTBC: 15

B. C. : 15  
VARBC : 34  
NA1 :  
    LO  
    LK-LP-ENV

NA2 :  
NA3 :  
NEXTBC: 16

B. C. : 16  
VARBC : 39  
NA1 :  
    HI  
    FL-EX-ENV

NA2 :  
NA3 :  
NEXTBC: 17

B. C. : 17  
VARBC : 43  
NA1 :  
    LO

NA2 :  
NA3 :  
NEXTBC: 18

B. C. : 18  
VARBC : 46  
NA1 :  
    HI

NA2 :  
NA3 :  
NEXTBC: 19

B. C. : 19  
 VARBC : 48  
 NA1 :  
 LO

NA2 :  
 NA3 :  
 NEXTBC: 20

B. C. : 20  
 VARBC : 50  
 NA1 :  
 HI

NA2 :  
 NA3 :  
 NEXTBC: 21

B. C. : 21  
 VARBC : 52  
 NA1 :  
 NA2 :  
 NA3 :  
 NEXTBC: 22

B. C. : 22  
 VARBC : 55  
 NA1 :  
 NA2 :  
 NA3 :  
 NEXTBC: 23

B. C. : 23  
 VARBC : 57  
 NA1 :  
 LO  
 CLOSED  
 BLOCKAGE  
 SHUT

NA2 :  
 NA3 :  
 NEXTBC: 24

B. C. : 24  
 VARBC : 62  
 NA1 :  
 HI  
 OPEN  
 WIDE-OPEN

NA2 :  
 FAIL-OPEN  
 NA3 :  
 NEXTBC: 25

B. C. : 25

VARBC : 67  
NA1 :  
- OPEN  
NA2 :  
NA3 :  
NEXTBC: 26

B. C. : 26  
VARBC : 78  
NA1 :  
- LO  
- LK-LP-ENV  
NA2 :  
NA3 :  
NEXTBC: 27

B. C. : 27  
VARBC : 75  
NA1 :  
- HI  
- FL-EX-ENV  
NA2 :  
NA3 :  
NEXTBC: 28

B. C. : 28  
VARBC : 80  
NA1 :  
- LO  
NA2 :  
NA3 :  
NEXTBC: 29

B. C. : 29  
VARBC : 83  
NA1 :  
- HI  
NA2 :  
NA3 :  
NEXTBC: 30

B. C. : 30  
VARBC : 85  
NA1 :  
- LO  
NA2 :  
NA3 :  
NEXTBC: 31

B. C. : 31  
VARBC : 87  
NA1 :  
- HI  
NA2 :  
NA3 :



NEXTBC: 32

B. C. : 32

VARBC : 89

NA1 :

NA2 :

NA3 :

NEXTBC: 33

B. C. : 33

VARBC : 92

NA1 :

NA2 :

NA3 :

NEXTBC: 34

## 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. C. : 39  
VARBC : 99  
NA1 :  
    HI  
    OPEN  
    FL-EX-ENV  
    WIDE-OPEN

NA2 :  
    FAIL-OPEN  
NA3 :  
NEXTBC: 40

B. C. : 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: 00000000  
 NAVA: NIL VARIABLE  
 FAULT: HI  
 NEXT: 2  
 TOP: 3  
 BACKG: NIL GATE  
 NEXTG: 1  
 UNN0: NIL UNIT  
 PROB : 0.0000

EVENT : 2  
 IDEN: T  
 UNID : 6 DUMMY-H  
 VAR: P  
 INOU: 00000000  
 NAVA: NIL VARIABLE  
 FAULT: LO  
 NEXT: 3  
 TOP: NIL EVENT  
 BACKG: 1  
 NEXTG: QM  
 UNN0: NIL UNIT  
 PROB : 0.0000

EVENT : 3  
 IDEN: M  
 UNID : 6 DUMMY-H  
 VAR: Q  
 INOU: 00000000  
 NAVA: NIL VARIABLE  
 FAULT: LO  
 NEXT: 4  
 TOP: 5  
 BACKG: NIL GATE  
 NEXTG: 2  
 UNN0: NIL UNIT  
 PROB : 0.0000

EVENT : 4  
 IDEN: T  
 UNID : 6 DUMMY-H  
 VAR: P  
 INOU: 00000000  
 NAVA: NIL VARIABLE  
 FAULT: HI  
 NEXT: 5  
 TOP: NIL EVENT

BACKG: 2  
NEXTG: 6M  
UNNØ: NIL UNIT  
PROB : 0.0000

EVENT : 5  
IDEN: M  
UNID : 6 DUMMY-H  
VAR: W  
INOU: 00000000  
NAVA: NIL VARIABLE  
FAULT: HI  
NEXT: 6  
TOP: 7  
BACKG: NIL GATE  
NEXTG: 3  
UNNØ: NIL UNIT  
PROB : 0.0000

EVENT : 6  
IDEN: B  
UNID : 6 DUMMY-H  
VAR: S  
INOU: 02000000  
NAVA: NIL VARIABLE  
FAULT: OTHER-CAU  
NEXT: 7  
TOP: NIL EVENT  
BACKG: 3  
NEXTG: NIL GATE  
UNNØ: NIL UNIT  
PROB : 0.0000

EVENT : 7  
IDEN: M  
UNID : 6 DUMMY-H  
VAR: W  
INOU: 02000000  
NAVA: NIL VARIABLE  
FAULT: LO  
NEXT: 8  
TOP: 9  
BACKG: NIL GATE  
NEXTG: 4  
UNNØ: NIL UNIT  
PROB : 0.0000

EVENT : 8  
IDEN: B  
UNID : 6 DUMMY-H  
VAR: S  
INOU: 02000000  
NAVA: NIL VARIABLE  
FAULT: OTHER-CAU  
NEXT: 9  
TOP: NIL EVENT  
BACKG: 4

NEXTG: NIL GATE  
 UNN0: NIL UNIT  
 PROB : 0.0000

EVENT : 9  
 IDEN: M  
 UNID : 6 DUMMY-H  
 VAR: T  
 INOU: 00000000  
 NAVA: NIL VARIABLE  
 FAULT: HI  
 NEXT: 10  
 TOP: 11  
 BACKG: NIL GATE  
 NEXTG: 5  
 UNN0: NIL UNIT  
 PROB : 0.0000

EVENT : 10  
 IDEN: B  
 UNID : 6 DUMMY-H  
 VAR: S  
 INOU: 00000000  
 NAVA: NIL VARIABLE  
 FAULT: OTHER-CAU  
 NEXT: 11  
 TOP: NIL EVENT  
 BACKG: 5  
 NEXTG: NIL GATE  
 UNN0: NIL UNIT  
 PROB : 0.0001

EVENT : 11  
 IDEN: M  
 UNID : 6 DUMMY-H  
 VAR: T  
 INOU: 00000000  
 NAVA: NIL VARIABLE  
 FAULT: LO  
 NEXT: 12  
 TOP: 13  
 BACKG: NIL GATE  
 NEXTG: 6  
 UNN0: NIL UNIT  
 PROB : 0.0000

EVENT : 12  
 IDEN: B  
 UNID : 6 DUMMY-H  
 VAR: S  
 INOU: 00000000  
 NAVA: NIL VARIABLE  
 FAULT: OTHER-CAU  
 NEXT: 13  
 TOP: NIL EVENT  
 BACKG: 6  
 NEXTG: NIL GATE

UNN0: NIL UNIT  
PROB : 0.0000

EVENT : 13  
IDEN: M  
UNID : 6 DUMMY-H  
VAR: X  
INO: 00000000  
NAVA: NIL VARIABLE  
FAULT: HI  
NEXT: 14  
TOP: 15  
BACKG: NIL GATE  
NEXTG: 7  
UNN0: NIL UNIT  
PROB : 0.0000

EVENT : 14  
IDEN: B  
UNID : 6 DUMMY-H  
VAR: S  
INO: 00000000  
NAVA: NIL VARIABLE  
FAULT: OTHER-CAU  
NEXT: 15  
TOP: NIL EVENT  
BACKG: 7  
NEXTG: NIL GATE  
UNN0: NIL UNIT  
PROB : 0.0000

EVENT : 15  
IDEN: M  
UNID : 6 DUMMY-H  
VAR: X  
INO: 00000000  
NAVA: NIL VARIABLE  
FAULT: LO  
NEXT: 16  
TOP: 17  
BACKG: NIL GATE  
NEXTG: 8  
UNN0: NIL UNIT  
PROB : 0.0000

EVENT : 16  
IDEN: B  
UNID : 6 DUMMY-H  
VAR: S  
INO: 00000000  
NAVA: NIL VARIABLE  
FAULT: OTHER-CAU  
NEXT: 17  
TOP: NIL EVENT  
BACKG: 8  
NEXTG: NIL GATE  
UNN0: 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  
UNN0: NIL UNIT  
PROB : 0.0000

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  
UNN0: NIL UNIT  
PROB : 0.0000

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: 10  
UNN0: NIL UNIT  
PROB : 0.0000

EVENT : 20  
IDEN: B  
UNID : 6 DUMMY-H  
VAR: S  
INOU: 02000000  
NAVA: NIL VARIABLE  
FAULT: OTHER-CAU  
NEXT: 21  
TOP: NIL EVENT  
BACKG: 10  
NEXTG: NIL GATE  
UNN0: NIL UNIT  
PROB : 0.0000



EVENT : 21  
 IDEN: M  
 UNID : 7 DUMMY-T  
 VAR: P  
 INOU: 10200000  
 NAVA: NIL VARIABLE  
 FAULT: HI  
 NEXT: 22  
 TOP: 23  
 BACKG: NIL GATE  
 NEXTG: 11  
 UNN0: NIL UNIT  
 PROB : 0.0000

EVENT : 22  
 IDEN: T  
 UNID : 7 DUMMY-T  
 VAR: Q  
 INOU: 10000000  
 NAVA: NIL VARIABLE  
 FAULT: HI  
 NEXT: 23  
 TOP: NIL EVENT  
 BACKG: 11  
 NEXTG: QM  
 UNN0: NIL UNIT  
 PROB : 0.0000

EVENT : 23  
 IDEN: M  
 UNID : 7 DUMMY-T  
 VAR: P  
 INOU: 10200000  
 NAVA: NIL VARIABLE  
 FAULT: LO  
 NEXT: 24  
 TOP: NIL EVENT  
 BACKG: NIL GATE  
 NEXTG: 12  
 UNN0: NIL UNIT  
 PROB : 0.0000

EVENT : 24  
 IDEN: T  
 UNID : 7 DUMMY-T  
 VAR: Q  
 INOU: 10000000  
 NAVA: NIL VARIABLE  
 FAULT: LO  
 NEXT: 25  
 TOP: NIL EVENT  
 BACKG: 12  
 NEXTG: QM  
 UNN0: NIL UNIT  
 PROB : 0.0000

EVENT : 25  
IDEN: M  
UNID : 9 PIPE  
VAR: Q  
INOU: 22000000  
NAVA: NIL VARIABLE  
FAULT: HI  
NEXT: 26  
TOP: 29  
BACKG: NIL GATE  
NEXTG: 13  
UNN0: NIL UNIT  
PROB : 0.0000

EVENT : 26  
IDEN: T  
UNID : 9 PIPE  
VAR: P  
INOU: 10000000  
NAVA: NIL VARIABLE  
FAULT: HI  
NEXT: 27  
TOP: NIL EVENT  
BACKG: 13  
NEXTG: QM  
UNN0: NIL UNIT  
PROB : 0.0000

EVENT : 27  
IDEN: T  
UNID : 9 PIPE  
VAR: P  
INOU: 00000000  
NAVA: NIL VARIABLE  
FAULT: LO  
NEXT: 28  
TOP: NIL EVENT  
BACKG: 13  
NEXTG: QM  
UNN0: NIL UNIT  
PROB : 0.0000

EVENT : 28  
IDEN: B  
UNID : 9 PIPE  
VAR: S  
INOU: 02000000  
NAVA: NIL VARIABLE  
FAULT: FL-EX-ENV  
NEXT: 29  
TOP: NIL EVENT  
BACKG: 13  
NEXTG: NIL GATE  
UNN0: NIL UNIT  
PROB : 0.0001

EVENT : 29  
IDEN: M  
UNID : 9 PIPE  
VAR: Q  
INOU: 00000000  
NAVA: NIL VARIABLE  
FAULT: LO  
NEXT: 30  
TOP: 34  
BACKG: NIL GATE  
NEXTG: 14  
UNN0: NIL UNIT  
PROB : 0.0000

EVENT : 30  
IDEN: T  
UNID : 9 PIPE  
VAR: P  
INOU: 10000000  
NAVA: NIL VARIABLE  
FAULT: LO  
NEXT: 31  
TOP: NIL EVENT  
BACKG: 14  
NEXTG: QM  
UNN0: NIL UNIT  
PROB : 0.0000

EVENT : 31  
IDEN: T  
UNID : 9 PIPE  
VAR: P  
INOU: 00000000  
NAVA: NIL VARIABLE  
FAULT: HI  
NEXT: 32  
TOP: NIL EVENT  
BACKG: 14  
NEXTG: QM  
UNN0: NIL UNIT  
PROB : 0.0000

EVENT : 32  
IDEN: B  
UNID : 9 PIPE  
VAR: S  
INOU: 02000000  
NAVA: NIL VARIABLE  
FAULT: BLOCKAGE  
NEXT: 33  
TOP: NIL EVENT  
BACKG: 14  
NEXTG: NIL GATE  
UNN0: NIL UNIT  
PROB : 0.0002

EVENT : 33

IDEN: B  
 UNID : 9 PIPE  
 VAR: S  
 INOU: 02000000  
 NAVA: NIL VARIABLE  
 FAULT: LK-LP-ENV  
 NEXT: 34 " "  
 TOP: NIL EVENT  
 BACKG: 14  
 NEXTG: NIL GATE  
 UNNO: NIL UNIT  
 PROB : 0.0025

EVENT : 34  
 IDEN: M  
 UNID : 9 PIPE  
 VAR: P  
 INOU: 10000000  
 NAVA: NIL VARIABLE  
 FAULT: HI  
 NEXT: 35  
 TOP: 39  
 BACKG: NIL GATE  
 NEXTG: 15  
 UNNO: NIL UNIT  
 PROB : 0.0000

EVENT : 35  
 IDEN: T  
 UNID : 9 PIPE  
 VAR: Q  
 INOU: 10000000  
 NAVA: NIL VARIABLE  
 FAULT: HI  
 NEXT: 36  
 TOP: NIL EVENT  
 BACKG: 15  
 NEXTG: QM  
 UNNO: NIL UNIT  
 PROB : 0.0000

EVENT : 36  
 IDEN: T  
 UNID : 9 PIPE  
 VAR: Q  
 INOU: 00000000  
 NAVA: NIL VARIABLE  
 FAULT: LO  
 NEXT: 37  
 TOP: NIL EVENT  
 BACKG: 15  
 NEXTG: QM  
 UNNO: NIL UNIT  
 PROB : 0.0000

EVENT : 37  
 IDEN: B

UNID : 9 PIPE  
 VAR: S  
 INOU: 02000000  
 NAVA: NIL VARIABLE  
 FAULT: BLOCKAGE  
 NEXT: 38  
 TOP: NIL EVENT  
 BACKG: 15  
 NEXTG: NIL GATE  
 UNN0: NIL UNIT  
 PROB : 0.0203

EVENT : 38  
 IDEN: B  
 UNID : 9 PIPE  
 VAR: S  
 INOU: 02000000  
 NAVA: NIL VARIABLE  
 FAULT: LK-HP-ENV  
 NEXT: 39  
 TOP: NIL EVENT  
 BACKG: 15  
 NEXTG: NIL GATE  
 UNN0: NIL UNIT  
 PROB : 0.0024

EVENT : 39  
 IDEN: M  
 UNID : 9 PIPE  
 VAR: P  
 INOU: 10000000  
 NAVA: NIL VARIABLE  
 FAULT: LO  
 NEXT: 40  
 TOP: 43  
 BACKG: NIL GATE  
 NEXTG: 16  
 UNN0: NIL UNIT  
 PROB : 0.0030

EVENT : 40  
 IDEN: T  
 UNID : 9 PIPE  
 VAR: Q  
 INOU: 00000000  
 NAVA: NIL VARIABLE  
 FAULT: HI  
 NEXT: 41  
 TOP: NIL EVENT  
 BACKG: 16  
 NEXTG: QM  
 UNN0: NIL UNIT  
 PROB : 2.0000

EVENT : 41  
 IDEN: T  
 UNID : 9 PIPE

VAR: Q  
INOU: 12020000  
NAVA: 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  
NAVA: 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  
NAVA: NIL VARIABLE  
FAULT: HI  
NEXT: 44  
TOP: 46  
BACKG: NIL GATE  
NEXTG: 17  
UNNO: NIL UNIT  
PROB: 0.0000

EVENT: 44  
IDEN: T  
UNID: 9 PIPE  
VAR: T  
INOU: 10000000  
NAVA: NIL VARIABLE  
FAULT: HI  
NEXT: 45  
TOP: NIL EVENT  
BACKG: 17  
NEXTG: QM  
UNNO: NIL UNIT  
PROB: 0.0000

EVENT: 45  
IDEN: B  
UNID: 9 PIPE  
VAR: S

INOU: 02000000  
NAVA: NIL VARIABLE  
FAULT: EXT-FIRE  
NEXT: 46  
TOP: NIL EVENT  
BACKG: 17  
NEXTG: NIL GATE  
UNN0: NIL UNIT  
PROB : 0.0001

EVENT : 46  
IDEN: M  
UNID : 9 PIPE  
VAR: T  
INOU: 00000000  
NAVA: NIL VARIABLE  
FAULT: LO  
NEXT: 47  
TOP: 48  
BACKG: NIL GATE  
NEXTG: 18  
UNN0: NIL UNIT  
PROB : 0.0000

EVENT : 47  
IDEN: T  
UNID : 9 PIPE  
VAR: T  
INOU: 10000000  
NAVA: NIL VARIABLE  
FAULT: LO  
NEXT: 48  
TOP: NIL EVENT  
BACKG: 18  
NEXTG: QM  
UNN0: NIL UNIT  
PROB : 0.0000

EVENT : 48  
IDEN: M  
UNID : 9 PIPE  
VAR: X  
INOU: 00000000  
NAVA: NIL VARIABLE  
FAULT: HI  
NEXT: 49  
TOP: 50  
BACKG: NIL GATE  
NEXTG: 19  
UNN0: NIL UNIT  
PROB : 0.0000

EVENT : 49  
IDEN: T  
UNID : 9 PIPE  
VAR: X  
INOU: 10000000

NAVA: NIL VARIABLE  
FAULT: HI  
NEXT: 50  
TOP: NIL EVENT  
BACKG: 19  
NEXTG: QM  
UNN0: NIL UNIT  
PROB : 0.0000

EVENT : 50  
IDEN: M  
UNID : 9 PIPE  
VAR: X  
INOU: 00000000  
NAVA: NIL VARIABLE  
FAULT: LO  
NEXT: 51  
TOP: 52  
BACKG: NIL GATE  
NEXTG: 20  
UNN0: NIL UNIT  
PROB : 0.0000

EVENT : 51  
IDEN: T  
UNID : 9 PIPE  
VAR: X  
INOU: 10000000  
NAVA: NIL VARIABLE  
FAULT: LO  
NEXT: 52  
TOP: NIL EVENT  
BACKG: 20  
NEXTG: QM  
UNN0: NIL UNIT  
PROB : 0.0000

EVENT : 52  
IDEN: M  
UNID : 9 PIPE  
VAR: Q  
INOU: 00000000  
NAVA: NIL VARIABLE  
FAULT: NO-FLOW  
NEXT: 53  
TOP: 55  
BACKG: NIL GATE  
NEXTG: 21  
UNN0: NIL UNIT  
PROB : 0.0000

EVENT : 53  
IDEN: T  
UNID : 9 PIPE  
VAR: Q  
INOU: 10000000  
NAVA: NIL VARIABLE



FAULT: NO-FLOW  
 NEXT: 54  
 TOP: NIL EVENT  
 BACKG: 21  
 NEXTG: QM  
 UNNØ: NIL UNIT  
 PROB : 0.0000

EVENT : 54  
 IDEN: B  
 UNID : 9 PIPE  
 VAR: S  
 INOU: 02000000  
 NAVA: NIL VARIABLE  
 FAULT: COMP-BLOC  
 NEXT: 55  
 TOP: NIL EVENT  
 BACKG: 21  
 NEXTG: NIL GATE  
 UNNØ: NIL UNIT  
 PROB : 0.0001

EVENT : 55  
 IDEN: M  
 UNID : 9 PIPE  
 VAR: Q  
 INOU: 00000000  
 NAVA: NIL VARIABLE  
 FAULT: GTØ  
 NEXT: 56  
 TOP: NIL EVENT  
 BACKG: NIL GATE  
 NEXTG: 22  
 UNNØ: NIL UNIT  
 PROB : 0.0000

EVENT : 56  
 IDEN: T  
 UNID : 9 PIPE  
 VAR: Q  
 INOU: 10200000  
 NAVA: NIL VARIABLE  
 FAULT: GTØ  
 NEXT: 57  
 TOP: NIL EVENT  
 BACKG: 22  
 NEXTG: QM  
 UNNØ: NIL UNIT  
 PROB : 0.0000

EVENT : 57  
 IDEN: M  
 UNID : 13 VALVE  
 VAR: Q  
 INOU: 00000000  
 NAVA: NIL VARIABLE  
 FAULT: HI

NEXT: 58  
TOP: 62  
BACKG: NIL GATE  
NEXTG: 23  
UNN0: NIL UNIT  
PROB : 0.0000

EVENT : 58  
IDEN: T  
UNID : 13 VALVE  
VAR: P  
INOU: 10000000  
NAVA: NIL VARIABLE  
FAULT: HI  
NEXT: 59  
TOP: NIL EVENT  
BACKG: 23  
NEXTG: QM  
UNN0: NIL UNIT  
PROB : 0.0000

EVENT : 59  
IDEN: T  
UNID : 13 VALVE  
VAR: P  
INOU: 00000000  
NAVA: NIL VARIABLE  
FAULT: LO  
NEXT: 60  
TOP: NIL EVENT  
BACKG: 23  
NEXTG: QM  
UNN0: NIL UNIT  
PROB : 0.0000

EVENT : 60  
IDEN: B  
UNID : 13 VALVE  
VAR: S  
INOU: 02000000  
NAVA: NIL VARIABLE  
FAULT: WIDE-OPEN  
NEXT: 61  
TOP: NIL EVENT  
BACKG: 23  
NEXTG: NIL GATE  
UNN0: NIL UNIT  
PROB : 0.0001

EVENT : 61  
IDEN: B  
UNID : 13 VALVE  
VAR: S  
INOU: 02000000  
NAVA: NIL VARIABLE  
FAULT: FL-EX-ENV  
NEXT: 62

TOP: NIL EVENT  
BACKG: 23  
NEXTG: NIL GATE  
UNNØ: NIL UNIT  
PROB : 0.0002

EVENT : 62  
IDEN: M  
UNID : 13 VALVE  
VAR: Q  
INOÜ: 00000000  
NAVA: NIL VARIABLE  
FAULT: LO  
NEXT: 63  
TOP: 67  
BACKG: NIL GATE  
NEXTG: 24  
UNNØ: NIL UNIT  
PROB : 0.0002

EVENT : 63  
IDEN: T  
UNID : 13 VALVE  
VAR: P  
INOÜ: 00000000  
NAVA: NIL VARIABLE  
FAULT: HI  
NEXT: 64  
TOP: NIL EVENT  
BACKG: 24  
NEXTG: QM  
UNNØ: NIL UNIT  
PROB : 0.0000

EVENT : 64  
IDEN: T  
UNID : 13 VALVE  
VAR: P  
INOÜ: 10000000  
NAVA: NIL VARIABLE  
FAULT: LO  
NEXT: 65  
TOP: NIL EVENT  
BACKG: 24  
NEXTG: QM  
UNNØ: NIL UNIT  
PROB : 0.0000

EVENT : 65  
IDEN: R  
UNID : 13 VALVE  
VAR: S  
INOÜ: 00000000  
NAVA: NIL VARIABLE  
FAULT: CLOSED  
NEXT: 66  
TOP: NIL EVENT

BACKG: 24  
NEXTG: QM  
UNNØ: NIL UNIT  
PROB : 0.0000

EVENT : 66  
IDEN: B  
UNID : 13 VALVE  
VAR: S  
INOU: 02000000  
NAVA: NIL VARIABLE  
FAULT: LK-LP-ENV  
NEXT: 67  
TOP: NIL EVENT  
BACKG: 24  
NEXTG: NIL GATE  
UNNØ: NIL UNIT  
PROB : 0.0000

EVENT : 67  
IDEN: M  
UNID : 13 VALVE  
VAR: S  
INOU: 02000000  
NAVA: NIL VARIABLE  
FAULT: CLOSED  
NEXT: 68  
TOP: 70  
BACKG: NIL GATE  
NEXTG: 25  
UNNØ: NIL UNIT  
PROB : 0.0000

EVENT : 68  
IDEN: B  
UNID : 13 VALVE  
VAR: S  
INOU: 02000000  
NAVA: NIL VARIABLE  
FAULT: BLOCKAGE  
NEXT: 69  
TOP: NIL EVENT  
BACKG: 25  
NEXTG: NIL GATE  
UNNØ: NIL UNIT  
PROB : 0.0004

EVENT : 69  
IDEN: B  
UNID : 13 VALVE  
VAR: S  
INOU: 02000000  
NAVA: NIL VARIABLE  
FAULT: SHUT  
NEXT: 70  
TOP: NIL EVENT  
BACKG: 25

NEXTG: NIL GATE  
UNNØ: NIL UNIT  
PROB : 0.0005

EVENT : 70  
IDEN: M  
UNID : 13 VALVE  
VAR: P  
INOÜ: 10020000  
NAVA: NIL VARIABLE  
FAULT: HI  
NEXT: 71  
TOP: 75  
BACKG: NIL GATE  
NEXTG: 26  
UNNØ: NIL UNIT  
PROB : 0.0000

EVENT : 71  
IDEN: T  
UNID : 13 VALVE  
VAR: Q  
INOÜ: 10020000  
NAVA: NIL VARIABLE  
FAULT: HI  
NEXT: 72  
TOP: NIL EVENT  
BACKG: 26  
NEXTG: QM  
UNNØ: NIL UNIT  
PROB : 0.0000

EVENT : 72  
IDEN: T  
UNID : 13 VALVE  
VAR: Q  
INOÜ: 00000000  
NAVA: NIL VARIABLE  
FAULT: LO  
NEXT: 73  
TOP: NIL EVENT  
BACKG: 26  
NEXTG: QM  
UNNØ: NIL UNIT  
PROB : 0.0000

EVENT : 73  
IDEN: R  
UNID : 13 VALVE  
VAR: S  
INOÜ: 02000000  
NAVA: NIL VARIABLE  
FAULT: CLOSED  
NEXT: 74  
TOP: NIL EVENT  
BACKG: 26  
NEXTG: QM

UNN0: NIL UNIT  
PROB : 0.0000

EVENT : 74  
IDEN: B  
UNID : 13 VALVE  
VAR: S  
INOU: 02000000  
NAVA: NIL VARIABLE  
FAULT: LK-HP-ENV  
NEXT: 75  
TOP: NIL EVENT  
BACKG: 26  
NEXTG: NIL GATE  
UNN0: NIL UNIT  
PROB : 0.0006

EVENT : 75  
IDEN: M  
UNID : 13 VALVE  
VAR: P  
INOU: 10000000  
NAVA: NIL VARIABLE  
FAULT: LO  
NEXT: 76  
TOP: 80  
BACKG: NIL GATE  
NEXTG: 27  
UNN0: NIL UNIT  
PROB : 0.0000

EVENT : 76  
IDEN: T  
UNID : 13 VALVE  
VAR: Q  
INOU: 02000000  
NAVA: NIL VARIABLE  
FAULT: HI  
NEXT: 77  
TOP: NIL EVENT  
BACKG: 27  
NEXTG: QM  
UNN0: NIL UNIT  
PROB : 0.0000

EVENT : 77  
IDEN: T  
UNID : 13 VALVE  
VAR: Q  
INOU: 10000000  
NAVA: NIL VARIABLE  
FAULT: LO  
NEXT: 78  
TOP: NIL EVENT  
BACKG: 27  
NEXTG: QM  
UNN0: NIL UNIT

PROB : 0.0000

EVENT : 78  
 IDEN: B  
 UNID : 13 VALVE  
 VAR: S  
 INOU: 02000000  
 NAVA: NIL VARIABLE  
 FAULT: LK-LP-ENV  
 NEXT: 79  
 TOP: NIL EVENT  
 BACKG: 27  
 NEXTG: NIL GATE  
 UNN0: NIL UNIT  
 PROB : 0.0007

EVENT : 79  
 IDEN: B  
 UNID : 13 VALVE  
 VAR: S  
 INOU: 02000000  
 NAVA: NIL VARIABLE  
 FAULT: WIDE-OPEN  
 NEXT: 80  
 TOP: NIL EVENT  
 BACKG: 27  
 NEXTG: NIL GATE  
 UNN0: NIL UNIT  
 PROB : 0.0008

EVENT : 80  
 IDEN: M  
 UNID : 13 VALVE  
 VAR: T  
 INOU: 02000000  
 NAVA: NIL VARIABLE  
 FAULT: HI  
 NEXT: 81  
 TOP: 83  
 BACKG: NIL GATE  
 NEXTG: 28  
 UNN0: NIL UNIT  
 PROB : 0.0000

EVENT : 81  
 IDEN: T  
 UNID : 13 VALVE  
 VAR: T  
 INOU: 10000000  
 NAVA: NIL VARIABLE  
 FAULT: HI  
 NEXT: 82  
 TOP: NIL EVENT  
 BACKG: 28  
 NEXTG: GM  
 UNN0: NIL UNIT  
 PROB : 0.0000

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 : 0.0000

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 : 0.0000



EVENT : 86  
 IDEN: T  
 UNID : 13 VALVE  
 VAR: X  
 INOU: 10000000  
 NAVA: NIL VARIABLE  
 FAULT: HI  
 NEXT: 87  
 TOP: NIL EVENT  
 BACKG: 30  
 NEXTG: QM  
 UNN0: NIL UNIT  
 PROB : 0.0000

EVENT : 87  
 IDEN: M  
 UNID : 13 VALVE  
 VAR: X  
 INOU: 00000000  
 NAVA: NIL VARIABLE  
 FAULT: LO  
 NEXT: 88  
 TOP: 89  
 BACKG: NIL GATE  
 NEXTG: 31  
 UNN0: NIL UNIT  
 PROB : 0.0000

EVENT : 88  
 IDEN: T  
 UNID : 13 VALVE  
 VAR: X  
 INOU: 10000000  
 NAVA: NIL VARIABLE  
 FAULT: LO  
 NEXT: 89  
 TOP: NIL EVENT  
 BACKG: 31  
 NEXTG: QM  
 UNN0: NIL UNIT  
 PROB : 0.0000

EVENT : 89  
 IDEN: M  
 UNID : 13 VALVE  
 VAR: Q  
 INOU: 00000000  
 NAVA: NIL VARIABLE  
 FAULT: NO-FLOW  
 NEXT: 90  
 TOP: 92  
 BACKG: NIL GATE  
 NEXTG: 32  
 UNN0: NIL UNIT  
 PROB : 0.0000

EVENT : 90  
IDEN: T  
UNID : 13 VALVE  
VAR: Q  
INOU: 10000000  
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: 02000000  
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: 00000000  
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: 10000000  
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: 00000000  
 NAVA: 4  
 FAULT: L0  
 NEXT: NIL EVENT  
 TOP: NIL EVENT  
 BACKG: NIL GATE  
 NEXTG: 34  
 UNN0: 4  
 PROB : 0.0000

EVENT : 95  
 IDEN: T  
 UNID : 9    PIPE  
 VAR: P  
 INOU: 10000000  
 NAVA: 3  
 FAULT: L0  
 NEXT: 110  
 TOP: NIL EVENT  
 BACKG: 34  
 NEXTG: 35  
 UNN0: 4  
 PROB : 0.0000

EVENT : 96  
 IDEN: T  
 UNID : 9    PIPE  
 VAR: Q  
 INOU: 10000000  
 NAVA: 3  
 FAULT: L0  
 NEXT: 109  
 TOP: NIL EVENT  
 BACKG: 35  
 NEXTG: 36  
 UNN0: 4  
 PROB : 0.0000

EVENT : 97  
 IDEN: T  
 UNID : 13    VALVE  
 VAR: P  
 INOU: 10000000  
 NAVA: 2  
 FAULT: L0  
 NEXT: 105  
 TOP: NIL EVENT

650  
BACKG: 36  
NEXTG: 37  
UNNØ: 3  
PROB : 0.0000

EVENT : 98  
IDEN: T  
UNID : 13 VALVE  
VAR: Q  
INOU: 10000000  
NAVA: 2  
FAULT: LO  
NEXT: 104  
TOP: NIL EVENT  
BACKG: 37  
NEXTG: 38  
UNNØ: 3  
PROB : 0.0000

EVENT : 99  
IDEN: T  
UNID : 9 PIPE  
VAR: P  
INOU: 10000000  
NAVA: 1  
FAULT: LO  
NEXT: 102  
TOP: NIL EVENT  
BACKG: 38  
NEXTG: 39  
UNNØ: 2  
PROB : 0.0000

EVENT : 100  
IDEN: T  
UNID : 9 PIPE  
VAR: Q  
INOU: 10000000  
NAVA: 1  
FAULT: LO  
NEXT: 101  
TOP: NIL EVENT  
BACKG: 39  
NEXTG: NIL GATE  
UNNØ: 2  
PROB : 0.0000

EVENT : 101  
IDEN: B  
UNID : 9 PIPE  
VAR: S  
INOU: 02000000  
NAVA: NIL VARIABLE  
FAULT: LK-LP-ENV  
NEXT: NIL EVENT  
TOP: NIL EVENT  
BACKG: 39

NEXTG: NIL GATE  
UNN: 2  
PROB : 0.0005

EVENT : 102  
IDEN: B  
UNID : 9 PIPE  
VAR: S  
INOU: 02000000  
NAVA: NIL VARIABLE  
FAULT: BLOCKAGE  
NEXT: 103  
TOP: NIL EVENT  
BACKG: 38  
NEXTG: NIL GATE  
UNN: 2  
PROB : 0.0002

EVENT : 103  
IDEN: B  
UNID : 9 PIPE  
VAR: S  
INOU: 02000000  
NAVA: NIL VARIABLE  
FAULT: LK-LP-ENV  
NEXT: NIL EVENT  
TOP: NIL EVENT  
BACKG: 38  
NEXTG: NIL GATE  
UNN: 2  
PROB : 0.0025

EVENT : 104  
IDEN: B  
UNID : 13 VALVE  
VAR: S  
INOU: 02000000  
NAVA: NIL VARIABLE  
FAULT: LK-LP-ENV  
NEXT: NIL EVENT  
TOP: NIL EVENT  
BACKG: 37  
NEXTG: NIL GATE  
UNN: 3  
PROB : 0.0007

EVENT : 105  
IDEN: R  
UNID : 13 VALVE  
VAR: S  
INOU: 02000000  
NAVA: NIL VARIABLE  
FAULT: CLOSED  
NEXT: 106  
TOP: NIL EVENT  
BACKG: 36  
NEXTG: 40

UNN0: 3  
PROB : 0.0000

EVENT : 126  
IDEN: B  
UNID : 13 VALVE  
VAR: S  
INOU: 02000000  
NAVA: NIL VARIABLE  
FAULT: BLOCKAGE  
NEXT: 127  
TOP: NIL EVENT  
BACKG: 40  
NEXTG: NIL GATE  
UNN0: 3  
PROB : 0.0004

EVENT : 127  
IDEN: B  
UNID : 13 VALVE  
VAR: S  
INOU: 02000000  
NAVA: NIL VARIABLE  
FAULT: SHUT  
NEXT: NIL EVENT  
TOP: NIL EVENT  
BACKG: 40  
NEXTG: NIL GATE  
UNN0: 3  
PROB : 0.0005

EVENT : 128  
IDEN: B  
UNID : 13 VALVE  
VAR: S  
INOU: 02000000  
NAVA: NIL VARIABLE  
FAULT: LK-LP-ENV  
NEXT: NIL EVENT  
TOP: NIL EVENT  
BACKG: 36  
NEXTG: NIL GATE  
UNN0: 3  
PROB : 0.0003

EVENT : 129  
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  
UNN0: 4

PROB : 0.0005

EVENT : 110  
IDEN: T --  
UNID : 9 PIPE  
VAR: P  
INOU: 00000000  
NAVA: 4  
FAULT: H1  
NEXT: 111  
TOP: NIL EVENT  
BACKG: 34  
NEXTG: NIL GATE  
UNN0: 4  
PROB : 0.0000

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  
UNN0: 4  
PROB : 0.0002

EVENT : 112  
IDEN: B --  
UNID : 9 PIPE  
VAR: S  
INOU: 02000000  
NAVA: NIL VARIABLE  
FAULT: LK-LP-ENV  
NEXT: NIL EVENT  
TOP: NIL EVENT  
BACKG: 34  
NEXTG: NIL GATE  
UNN0: 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-LP-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) : GT0  
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: S  
BACKE: 1  
NEXTE: 2  
NEGATE: 2  
EC : 1

GATE : 2  
CLASS: 0 0  
EXTRA: S  
BACKE: 3  
NEXTE: 4  
NEGATE: 3  
EC : 2

GATE : 3  
CLASS: 0 0  
EXTRA: S  
BACKE: 5  
NEXTE: 6  
NEGATE: 4  
EC : 3

GATE : 4  
CLASS: 0 0  
EXTRA: S  
BACKE: 7  
NEXTE: 8  
NEGATE: 5  
EC : 4

GATE : 5  
CLASS: 0 0  
EXTRA: S  
BACKE: 9  
NEXTE: 10  
NEGATE: 6  
EC : 5

GATE : 6  
CLASS: 0 0  
EXTRA: S  
BACKE: 11  
NEXTE: 12  
NEGATE: 7  
EC : 6

GATE : 7  
CLASS: 0 0  
EXTRA: S  
BACKE: 13  
NEXTE: 14  
NEGATE: 8  
EC : 7

GATE : 8  
CLASS: 0 0  
EXTRA: S  
BACKE: 15  
NEXTE: 16  
NEGATE: 9  
EC : 8

GATE : 9  
CLASS: 0 0  
EXTRA: S  
BACKE: 17  
NEXTE: 18  
NEGATE: 10  
EC : 9

GATE : 10  
CLASS: 0 0  
EXTRA: S  
BACKE: 19  
NEXTE: 20  
NEGATE: 11  
EC : 10

GATE : 11  
CLASS: 0 0  
EXTRA: S  
BACKE: 21  
NEXTE: 22  
NEGATE: 12  
EC : 11

GATE : 12  
CLASS: 0 0  
EXTRA: S  
BACKE: 23  
NEXTE: 24  
NEGATE: 13  
EC : 12

GATE : 13  
CLASS: 0 0  
EXTRA: S  
BACKE: 25  
NEXTE: 26  
NEGATE: 14  
EC : 13

GATE : 14  
CLASS: 0 0  
EXTRA: S  
BACKE: 29  
NEXTE: 30  
NEGATE: 15  
BC : 14

GATE : 15  
CLASS: 0 0  
EXTRA: S  
BACKE: 34  
NEXTE: 35  
NEGATE: 16  
BC : 15

GATE : 16  
CLASS: 0 0  
EXTRA: S  
BACKE: 39  
NEXTE: 40  
NEGATE: 17  
BC : 16

GATE : 17  
CLASS: 0 0  
EXTRA: S  
BACKE: 43  
NEXTE: 44  
NEGATE: 18  
BC : 17

GATE : 18  
CLASS: 0 0  
EXTRA: S  
BACKE: 46  
NEXTE: 47  
NEGATE: 19  
BC : 18

GATE : 19  
CLASS: 0 0  
EXTRA: S  
BACKE: 48  
NEXTE: 49  
NEGATE: 20  
BC : 19

GATE : 20  
CLASS: 0 0  
EXTRA: S  
BACKE: 50  
NEXTE: 51

NEGATE: 21  
BC : 20

GATE : 21  
CLASS: 0 0  
EXTRA: S  
BACKE: 52  
NEXTE: 53  
NEGATE: 22  
BC : 21

GATE : 22  
CLASS: 0 0  
EXTRA: S  
BACKE: 55  
NEXTE: 56  
NEGATE: 23  
BC : 22

GATE : 23  
CLASS: 0 0  
EXTRA: S  
BACKE: 57  
NEXTE: 58  
NEGATE: 24  
BC : 23

GATE : 24  
CLASS: 0 0  
EXTRA: S  
BACKE: 62  
NEXTE: 63  
NEGATE: 25  
BC : 24

GATE : 25  
CLASS: 0 0  
EXTRA: S  
BACKE: 67  
NEXTE: 68  
NEGATE: 26  
BC : 25

GATE : 26  
CLASS: 0 0  
EXTRA: S  
BACKE: 70  
NEXTE: 71  
NEGATE: 27  
BC : 26

GATE : 27  
CLASS: 0 0  
EXTRA: S

BACKE: 75  
NEXTE: 76  
NEGATE: 28  
EC : 27

GATE : 28  
CLASS: 0 0  
EXTRA: S  
BACKE: 80  
NEXTE: 81  
NEGATE: 29  
EC : 28

GATE : 29  
CLASS: 0 0  
EXTRA: S  
BACKE: 83  
NEXTE: 84  
NEGATE: 30  
EC : 29

GATE : 30  
CLASS: 0 0  
EXTRA: S  
BACKE: 85  
NEXTE: 86  
NEGATE: 31  
EC : 30

GATE : 31  
CLASS: 0 0  
EXTRA: S  
BACKE: 87  
NEXTE: 88  
NEGATE: 32  
EC : 31

GATE : 32  
CLASS: 0 0  
EXTRA: S  
BACKE: 89  
NEXTE: 90  
NEGATE: 33  
EC : 32

GATE : 33  
CLASS: 0 0  
EXTRA: S  
BACKE: 92  
NEXTE: 93  
NEGATE: 34  
EC : 33

## ARRAY OF GATES

\*\*\*\*\*

GATE : 34  
 CLASS: 0 0  
 EXTRA: S  
 BACKE: 94  
 NEXTE: 95  
 NEGATE: 35  
 EC : 34

GATE : 35  
 CLASS: 0 0  
 EXTRA: S  
 BACKE: 95  
 NEXTE: 96  
 NEGATE: 36  
 EC : 35

GATE : 36  
 CLASS: 0 0  
 EXTRA: S  
 BACKE: 96  
 NEXTE: 97  
 NEGATE: 37  
 EC : 36

GATE : 37  
 CLASS: 0 0  
 EXTRA: S  
 BACKE: 97  
 NEXTE: 98  
 NEGATE: 38  
 EC : 37

GATE : 38  
 CLASS: 0 0  
 EXTRA: S  
 BACKE: 98  
 NEXTE: 99  
 NEGATE: 39  
 EC : 38

GATE : 39  
 CLASS: 0 0  
 EXTRA: S  
 BACKE: 99  
 NEXTE: 100  
 NEGATE: 40  
 EC : 39

GATE : 40  
CLASS: 0 0  
EXTRA: S  
BACKE: 105  
NEXTE: 106  
NEGATE: 41  
EC : 40



## ARRAY OF DATA FOR VARIABLES

\*\*\*\*\*

DAT : 1  
HL : 100  
LL : 0  
PRVA : 0  
ACTVA : 0  
PRIO : 1  
PREX : S

DAT : 2  
HL : 100  
LL : 0  
PRVA : 0  
ACTVA : 0  
PRIO : 2  
PREX : S

DAT : 3  
HL : 0  
LL : 0  
PRVA : 0  
ACTVA : 0  
PRIO : 32  
PREX : S

DAT : 4  
HL : 0  
LL : 0  
PRVA : 0  
ACTVA : 0  
PRIO : 32  
PREX : S

DAT : 5  
HL : 0  
LL : 0  
PRVA : 0  
ACTVA : 0  
PRIO : 32  
PREX : S

DAT : 6  
HL : 0  
LL : 0  
PRVA : 0  
ACTVA : 0  
PRIO : 32  
PREX : S

DAT : 7  
HL : 0  
LL : 0  
PRVA : 0  
ACTVA : 0  
PRIO : 32  
PREX : S

DAT : 8  
HL : 0  
LL : 0  
PRVA : 0  
ACTVA : 0  
PRIO : 32  
PREX : S

DAT : 9  
HL : 0  
LL : 0  
PRVA : 0  
ACTVA : 0  
PRIO : 32  
PREX : S

DAT : 10  
HL : 0  
LL : 0  
PRVA : 0  
ACTVA : 0  
PRIO : 32  
PREX : S

DAT : 11  
HL : 0  
LL : 0  
PRVA : 0  
ACTVA : 0  
PRIO : 32  
PREX : S

DAT : 12  
HL : 0  
LL : 0  
PRVA : 0  
ACTVA : 0  
PRIO : 32  
PREX : S

DAT : 13  
HL : 0  
LL : 0  
PRVA : 0  
ACTVA : 0  
PRIO : 32  
PREX : S

DAT : 14  
HL : 0  
LL : 0  
PRVA : 0  
ACTVA : 0  
PRIO : 32  
PREX : S

DAT : 15  
HL : 0  
LL : 0  
PRVA : 0  
ACTVA : 0  
PRIO : 32  
PREX : S

DAT : 16  
HL : 0  
LL : 0  
PRVA : 0  
ACTVA : 0  
PRIO : 32  
PREX : S

DAT : 17  
HL : 0  
LL : 0  
PRVA : 0  
ACTVA : 0  
PRIO : 32  
PREX : S

DAT : 18  
HL : 0  
LL : 0  
PRVA : 0  
ACTVA : 0  
PRIO : 32  
PREX : S

DAT : 19  
HL : 0  
LL : 0  
PRVA : 0  
ACTVA : 0  
PRIO : 32  
PREX : S

DAT : 20  
HL : 0  
LL : 0  
PRVA : 0  
ACTVA : 0

PRI0 : 32  
PREX : S

DAT : 21  
HL : 0  
LL : 0  
PRVA : 0  
ACTVA : 0  
PRI0 : 32  
PREX : S

DAT : 22  
HL : 0  
LL : 0  
PRVA : 0  
ACTVA : 0  
PRI0 : 32  
PREX : S

DAT : 23  
HL : 0  
LL : 0  
PRVA : 0  
ACTVA : 0  
PRI0 : 32  
PREX : S

DAT : 24  
HL : 0  
LL : 0  
PRVA : 0  
ACTVA : 0  
PRI0 : 32  
PREX : S

DAT : 25  
HL : 0  
LL : 0  
PRVA : 0  
ACTVA : 0  
PRI0 : 32  
PREX : S

DAT : 26  
HL : 0  
LL : 0  
PRVA : 0  
ACTVA : 0  
PRI0 : 32  
PREX : S

DAT : 27  
HL : 0  
LL : 0

PRVA : 0  
ACTVA : 0  
PRIO : 32  
PREX : S

DAT : 28  
HL : 0  
LL : 0  
PRVA : 0  
ACTVA : 0  
PRIO : 32  
PREX : S

DAT : 29  
HL : 0  
LL : 0  
PRVA : 0  
ACTVA : 0  
PRIO : 32  
PREX : S

DAT : 30  
HL : 0  
LL : 0  
PRVA : 0  
ACTVA : 0  
PRIO : 32  
PREX : S

## ARRAY OF STREAMS

\*\*\*\*\*

STREAM: 1  
 SI0: 0  
 EIO: 0 1  
 FROMU: 1  
 TOUN: 2  
 NAVAR: 1  
 NEXTS: NIL STREAM

STREAM: 2  
 SI0: 1  
 EIO: 0 0  
 FROMU: 1  
 TOUN: 2  
 NAVAR: NIL VARIABLE  
 NEXTS: NIL STREAM

STREAM: 3  
 SI0: 0  
 EIO: 0 1  
 FROMU: 2  
 TOUN: 3  
 NAVAR: 3  
 NEXTS: NIL STREAM

STREAM: 4  
 SI0: 1  
 EIO: 0 0  
 FROMU: 2  
 TOUN: 3  
 NAVAR: NIL VARIABLE  
 NEXTS: NIL STREAM

STREAM: 5  
 SI0: 0  
 EIO: 0 1  
 FROMU: 3  
 TOUN: 4  
 NAVAR: 5  
 NEXTS: NIL STREAM

STREAM: 6  
 SI0: 1  
 EIO: 0 0  
 FROMU: 3  
 TOUN: 4  
 NAVAR: NIL VARIABLE  
 NEXTS: NIL STREAM

STREAM: 7  
SI0: 0  
EIO: 0 1  
FROMU: 4  
TOUN: 5  
NAVAR: 7  
NEXTS: NIL STREAM

STREAM: 8  
SI0: 1  
EIO: 0 0  
FROMU: 4  
TOUN: 5  
NAVAR: NIL VARIABLE  
NEXTS: NIL STREAM

STREAM: 9  
SI0: 2  
EIO: 2  
FROMU: NIL UNIT  
TOUN: NIL UNIT  
NAVAR: NIL VARIABLE  
NEXTS: 10

STREAM: 10  
SI0: 2  
EIO: 2  
FROMU: NIL UNIT  
TOUN: NIL UNIT  
NAVAR: NIL VARIABLE  
NEXTS: 11

STREAM: 11  
SI0: 2  
EIO: 2  
FROMU: NIL UNIT  
TOUN: NIL UNIT  
NAVAR: NIL VARIABLE  
NEXTS: 12

STREAM: 12  
SI0: 2  
EIO: 2  
FROMU: NIL UNIT  
TOUN: NIL UNIT  
NAVAR: NIL VARIABLE  
NEXTS: 13

STREAM: 13  
SI0: 2  
EIO: 2  
FROMU: NIL UNIT  
TOUN: NIL UNIT  
NAVAR: NIL VARIABLE  
NEXTS: 14

STREAM: 14  
SIØ: 2  
EIO: 2  
FROMU: NIL UNIT  
TOUN: NIL UNIT  
NAVAR: NIL VARIABLE  
NEXTS: 15

STREAM: 15  
SIØ: 2  
EIO: 2  
FROMU: NIL UNIT  
TOUN: NIL UNIT  
NAVAR: NIL VARIABLE  
NEXTS: 16

STREAM: 16  
SIØ: 2  
EIO: 2  
FROMU: NIL UNIT  
TOUN: NIL UNIT  
NAVAR: NIL VARIABLE  
NEXTS: 17

STREAM: 17  
SIØ: 2  
EIO: 2  
FROMU: NIL UNIT  
TOUN: NIL UNIT  
NAVAR: NIL VARIABLE  
NEXTS: 18

STREAM: 18  
SIØ: 2  
EIO: 2  
FROMU: NIL UNIT  
TOUN: NIL UNIT  
NAVAR: NIL VARIABLE  
NEXTS: 19

STREAM: 19  
SIØ: 2  
EIO: 2  
FROMU: NIL UNIT  
TOUN: NIL UNIT  
NAVAR: NIL VARIABLE  
NEXTS: 20

STREAM: 20  
SIØ: 2  
EIO: 2  
FROMU: NIL UNIT  
TOUN: NIL UNIT



NAVAR: NIL VARIABLE  
NEXTS: 21

STREAM: 21  
SIØ: Z  
EIO: Z  
FROMU: NIL UNIT  
TOUN: NIL UNIT  
NAVAR: NIL VARIABLE  
NEXTS: 22

STREAM: 22  
SIØ: Z  
EIO: Z  
FROMU: NIL UNIT  
TOUN: NIL UNIT  
NAVAR: NIL VARIABLE  
NEXTS: 23

STREAM: 23  
SIØ: Z  
EIO: Z  
FROMU: NIL UNIT  
TOUN: NIL UNIT  
NAVAR: NIL VARIABLE  
NEXTS: 24

STREAM: 24  
SIØ: Z  
EIO: Z  
FROMU: NIL UNIT  
TOUN: NIL UNIT  
NAVAR: NIL VARIABLE  
NEXTS: 25

STREAM: 25  
SIØ: Z  
EIO: Z  
FROMU: NIL UNIT  
TOUN: NIL UNIT  
NAVAR: NIL VARIABLE  
NEXTS: 26

STREAM: 26  
SIØ: Z  
EIO: Z  
FROMU: NIL UNIT  
TOUN: NIL UNIT  
NAVAR: NIL VARIABLE  
NEXTS: 27

STREAM: 27  
SIØ: Z  
EIO: Z

FROMU: NIL UNIT  
TOUN: NIL UNIT  
NAVAR: NIL VARIABLE  
NEXTS: 28

STREAM: 28  
SIØ: Z  
EIO: Z  
FROMU: NIL UNIT  
TOUN: NIL UNIT  
NAVAR: NIL VARIABLE  
NEXTS: 29

STREAM: 29  
SIØ: Z  
EIO: Z  
FROMU: NIL UNIT  
TOUN: NIL UNIT  
NAVAR: NIL VARIABLE  
NEXTS: 30

STREAM: 30  
SIØ: Z  
EIO: Z  
FROMU: NIL UNIT  
TOUN: NIL UNIT  
NAVAR: NIL VARIABLE  
NEXTS: 31

STREAM: 31  
SIØ: Z  
EIO: Z  
FROMU: NIL UNIT  
TOUN: NIL UNIT  
NAVAR: NIL VARIABLE  
NEXTS: 32

STREAM: 32  
SIØ: Z  
EIO: Z  
FROMU: NIL UNIT  
TOUN: NIL UNIT  
NAVAR: NIL VARIABLE  
NEXTS: 33

STREAM: 33  
SIØ: Z  
EIO: Z  
FROMU: NIL UNIT  
TOUN: NIL UNIT  
NAVAR: NIL VARIABLE  
NEXTS: 34

STREAM: 34

SIØ: Z  
EIO: Z  
FROMU: NIL UNIT  
TOUN: NIL UNIT  
NAVAR: NIL VARIABLE  
NEXTS: 35

STREAM: 35  
SIØ: Z  
EIO: Z  
FROMU: NIL UNIT  
TOUN: NIL UNIT  
NAVAR: NIL VARIABLE  
NEXTS: 36

STREAM: 36  
SIØ: Z  
EIO: Z  
FROMU: NIL UNIT  
TOUN: NIL UNIT  
NAVAR: NIL VARIABLE  
NEXTS: 37

STREAM: 37  
SIØ: Z  
EIO: Z  
FROMU: NIL UNIT  
TOUN: NIL UNIT  
NAVAR: NIL VARIABLE  
NEXTS: 38

STREAM: 38  
SIØ: Z  
EIO: Z  
FROMU: NIL UNIT  
TOUN: NIL UNIT  
NAVAR: NIL VARIABLE  
NEXTS: 39

STREAM: 39  
SIØ: Z  
EIO: Z  
FROMU: NIL UNIT  
TOUN: NIL UNIT  
NAVAR: NIL VARIABLE  
NEXTS: 40

STREAM: 40  
SIØ: Z  
EIO: Z  
FROMU: NIL UNIT  
TOUN: NIL UNIT  
NAVAR: NIL VARIABLE  
NEXTS: 41

STREAM: 41  
SIØ: Z  
EIO: Z  
FROMU: NIL UNIT  
TOUN: NIL UNIT  
NAVAR: NIL VARIABLE  
NEXTS: 42

STREAM: 42  
SIØ: Z  
EIO: Z  
FROMU: NIL UNIT  
TOUN: NIL UNIT  
NAVAR: NIL VARIABLE  
NEXTS: 43

STREAM: 43  
SIØ: Z  
EIO: Z  
FROMU: NIL UNIT  
TOUN: NIL UNIT  
NAVAR: NIL VARIABLE  
NEXTS: 44

STREAM: 44  
SIØ: Z  
EIO: Z  
FROMU: NIL UNIT  
TOUN: NIL UNIT  
NAVAR: NIL VARIABLE  
NEXTS: 45

STREAM: 45  
SIØ: Z  
EIO: Z  
FROMU: NIL UNIT  
TOUN: NIL UNIT  
NAVAR: NIL VARIABLE  
NEXTS: 46

STREAM: 46  
SIØ: Z  
EIO: Z  
FROMU: NIL UNIT  
TOUN: NIL UNIT  
NAVAR: NIL VARIABLE  
NEXTS: 47

STREAM: 47  
SIØ: Z  
EIO: Z  
FROMU: NIL UNIT  
TOUN: NIL UNIT  
NAVAR: NIL VARIABLE

NEXTS: 48

STREAM: 48

SIØ: 2

EIO: 2

FROMU: NIL UNIT

TOUN: NIL UNIT

NAVAR: NIL VARIABLE

NEXTS: 49

STREAM: 49

SIØ: 2

EIO: 2

FROMU: NIL UNIT

TOUN: NIL UNIT

NAVAR: NIL VARIABLE

NEXTS: 50

STREAM: 50

SIØ: 2

EIO: 2

FROMU: NIL UNIT

TOUN: NIL UNIT

NAVAR: NIL VARIABLE

NEXTS: NIL NEXT STREAM

## ARRAY TO LOCATE TOP EVENTS

\*\*\*\*\*

LOT: 1  
TYPE: DUMMY-H  
TEX: S  
FIRSEL: 1

LOT: 2  
TYPE: DUMMY-T  
TEX: S  
FIRSEL: 21

LOT: 3  
TYPE: PIPE  
TEX: S  
FIRSEL: 25

LOT: 4  
TYPE: VALVE  
TEX: S  
FIRSEL: 57

LOT: 5  
TYPE: S  
TEX: S  
FIRSEL: NIL EVENT

LOT: 6  
TYPE: S  
TEX: S  
FIRSEL: NIL EVENT

LOT: 7  
TYPE: S  
TEX: S  
FIRSEL: NIL EVENT

LOT: 8  
TYPE: S  
TEX: S  
FIRSEL: NIL EVENT

LOT: 9  
TYPE: S  
TEX: S  
FIRSEL: NIL EVENT

LOT: 10  
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

\*\*\*\*\*

TYPE OF UNIT (1) : CENT-PUMP

TYPE OF UNIT (2) : CLOSED-TK

TYPE OF UNIT (3) : CNTRLV-SC

TYPE OF UNIT (4) : CNTRL-VAL

TYPE OF UNIT (5) : CNTRROLLER

TYPE OF UNIT (6) : DUMMY-H

TYPE OF UNIT (7) : DUMMY-T

TYPE OF UNIT (8) : HEAT-EX

TYPE OF UNIT (9) : PIPE

TYPE OF UNIT (10) : SENSOR-Q

TYPE OF UNIT (11) : SENSOR-P

TYPE OF UNIT (12) : SENSOR-T

TYPE OF UNIT (13) : VALVE

TYPE OF UNIT (14) :

TYPE OF UNIT (15) :



## ARRAY OF UNITS

\*\*\*\*\*

UNIT: 1  
UNUM: 1  
TY: DUMMY-H  
TYE: S  
INST: NILINS  
OUTST: 1  
NEXU: NIL UNIT

UNIT: 2  
UNUM: 2  
TY: PIPE  
TYE: S  
INST: 2  
OUTST: 3  
NEXU: NIL UNIT

UNIT: 3  
UNUM: 3  
TY: VALVE  
TYE: S  
INST: 4  
OUTST: 5  
NEXU: NIL UNIT

UNIT: 4  
UNUM: 4  
TY: PIPE  
TYE: S  
INST: 6  
OUTST: 7  
NEXU: NIL UNIT

UNIT: 5  
UNUM: 5  
TY: DUMMY-T  
TYE: S  
INST: 8  
OUTST: NILOUT  
NEXU: NIL UNIT

UNIT: 6  
UNUM: 6  
TY: S  
TYE: S  
INST: NIL STREAM  
OUTST: NIL STREAM  
NEXU: 7

UNIT: 7  
UNUM: 0  
TY: S  
TYE: S  
INST: NIL STREAM  
OUTST: NIL STREAM  
NEXU: 8

UNIT: 8  
UNUM: 0  
TY: S  
TYE: S  
INST: NIL STREAM  
OUTST: NIL STREAM  
NEXU: 9

UNIT: 9  
UNUM: 0  
TY: S  
TYE: S  
INST: NIL STREAM  
OUTST: NIL STREAM  
NEXU: 10

UNIT: 10  
UNUM: 0  
TY: S  
TYE: S  
INST: NIL STREAM  
OUTST: NIL STREAM  
NEXU: 11

UNIT: 11  
UNUM: 0  
TY: S  
TYE: S  
INST: NIL STREAM  
OUTST: NIL STREAM  
NEXU: 12

UNIT: 12  
UNUM: 0  
TY: S  
TYE: S  
INST: NIL STREAM  
OUTST: NIL STREAM  
NEXU: 13

UNIT: 13  
UNUM: 0  
TY: S  
TYE: S  
INST: NIL STREAM  
OUTST: NIL STREAM  
NEXU: 14

UNIT: 14  
UNUM: 0  
TY: S  
TYE: S  
INST: NIL STREAM  
OUTST: NIL STREAM  
NEXU: 15

UNIT: 15  
UNUM: 0  
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 : NIL DAT

VAR: 2  
 NAMEF: Q  
 NAMET: Q  
 MESU : N  
 EX : 3  
 NOVA: 1  
 BACKS: 1  
 NEXVAR: NIL VARIABLE  
 VALUES : NIL DAT

VAR: 3  
 NAMEF: P  
 NAMET: P  
 MESU : N  
 EX : 3  
 NOVA: 2  
 BACKS: 3  
 NEXVAR: 4  
 VALUES : NIL DAT

VAR: 4  
 NAMEF: Q  
 NAMET: Q  
 MESU : N  
 EX : 3  
 NOVA: 2  
 BACKS: 3  
 NEXVAR: NIL VARIABLE  
 VALUES : NIL DAT

VAR: 5  
 NAMEF: P  
 NAMET: P  
 MESU : Y  
 EX : 3  
 NOVA: 3  
 BACKS: 5  
 NEXVAR: 6  
 VALUES : 1

VAR: 6  
NAMEF: Q  
NAMET: Q  
MESU : N  
EX : 3  
NOVA: 3  
BACKS: 5  
NEXVAR: NIL VARIABLE  
VALUES : NILDAT

VAR: 7  
NAMEF: P  
NAMET: P  
MESU : N  
EX : 3  
NOVA: 4  
BACKS: 7  
NEXVAR: 8  
VALUES : NILDAT

VAR: 8  
NAMEF: Q  
NAMET: Q  
MESU : Y  
EX : 3  
NOVA: 4  
BACKS: 7  
NEXVAR: NIL VARIABLE  
VALUES : 2

VAR: 9  
NAMEF: S  
NAMET: S  
MESU : S  
EX : 2  
NOVA: 0  
BACKS: NIL STREAM  
NEXVAR: 10  
VALUES : NILDAT

VAR: 10  
NAMEF: S  
NAMET: S  
MESU : S  
EX : 2  
NOVA: 0  
BACKS: NIL STREAM  
NEXVAR: 11  
VALUES : NILDAT

VAR: 11  
NAMEF: S  
NAMET: S  
MESU : S  
EX : 2

NOVA: 0  
BACKS: NIL STREAM  
NEXVAR: 12  
VALUES : NILDAT

VAR: 12  
NAMEF: S  
NAMET: S  
MESU : S  
EX :Z  
NOVA: 0  
BACKS: NIL STREAM  
NEXVAR: 13  
VALUES : NILDAT

VAR: 13  
NAMEF: S  
NAMET: S  
MESU : S  
EX :Z  
NOVA: 0  
BACKS: NIL STREAM  
NEXVAR: 14  
VALUES : NILDAT

VAR: 14  
NAMEF: S  
NAMET: S  
MESU : S  
EX :Z  
NOVA: 0  
BACKS: NIL STREAM  
NEXVAR: 15  
VALUES : NILDAT

VAR: 15  
NAMEF: S  
NAMET: S  
MESU : S  
EX :Z  
NOVA: 0  
BACKS: NIL STREAM  
NEXVAR: 16  
VALUES : NILDAT

VAR: 16  
NAMEF: S  
NAMET: S  
MESU : S  
EX :Z  
NOVA: 0  
BACKS: NIL STREAM  
NEXVAR: 17  
VALUES : NILDAT

VAR: 17  
NAMEF: S  
NAMET: S  
MESU : S  
EX :Z  
NOVA: 0  
BACKS: NIL STREAM  
NEXVAR: 18  
VALUES : NILDAT

VAR: 18  
NAMEF: S  
NAMET: S  
MESU : S  
EX :Z  
NOVA: 0  
BACKS: NIL STREAM  
NEXVAR: 19  
VALUES : NILDAT

VAR: 19  
NAMEF: S  
NAMET: S  
MESU : S  
EX :Z  
NOVA: 0  
BACKS: NIL STREAM  
NEXVAR: 20  
VALUES : NILDAT

VAR: 20  
NAMEF: S  
NAMET: S  
MESU : S  
EX :Z  
NOVA: 2  
BACKS: NIL STREAM  
NEXVAR: 21  
VALUES : NILDAT

VAR: 21  
NAMEF: S  
NAMET: S  
MESU : S  
EX :Z  
NOVA: 0  
BACKS: NIL STREAM  
NEXVAR: 22  
VALUES : NILDAT

VAR: 22  
NAMEF: S  
NAMET: S  
MESU : S  
EX :Z  
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



NAMEF: S  
NAMET: S  
MESU : S  
EX :Z  
NOVA: 0  
BACKS: NIL STREAM  
NEXVAR: 29  
VALUES : NILDAT

VAR: 29  
NAMEF: S  
NAMET: S  
MESU : S  
EX :Z  
NOVA: 0  
BACKS: NIL STREAM  
NEXVAR: 30  
VALUES : NILDAT

VAR: 30  
NAMEF: S  
NAMET: S  
MESU : S  
EX :Z  
NOVA: 0  
BACKS: NIL STREAM  
NEXVAR: 31  
VALUES : NILDAT

VAR: 31  
NAMEF: S  
NAMET: S  
MESU : S  
EX :Z  
NOVA: 0  
BACKS: NIL STREAM  
NEXVAR: 32  
VALUES : NILDAT

VAR: 32  
NAMEF: S  
NAMET: S  
MESU : S  
EX :Z  
NOVA: 0  
BACKS: NIL STREAM  
NEXVAR: 33  
VALUES : NILDAT

VAR: 33  
NAMEF: S  
NAMET: S  
MESU : S  
EX :Z  
NOVA: 0  
BACKS: NIL STREAM

NEXVAR: 34  
VALUES : NILDAT

VAR: 34  
NAMEF: S  
NAMET: S  
MESU : S  
EX :Z  
NOVA: Ø  
BACKS: NIL STREAM  
NEXVAR: 35  
VALUES : NILDAT

VAR: 35  
NAMEF: S  
NAMET: S  
MESU : S  
EX :Z  
NOVA: Ø  
BACKS: NIL STREAM  
NEXVAR: 36  
VALUES : NILDAT

VAR: 36  
NAMEF: S  
NAMET: S  
MESU : S  
EX :Z  
NOVA: Ø  
BACKS: NIL STREAM  
NEXVAR: 37  
VALUES : NILDAT

VAR: 37  
NAMEF: S  
NAMET: S  
MESU : S  
EX :Z  
NOVA: Ø  
BACKS: NIL STREAM  
NEXVAR: 38  
VALUES : NILDAT

VAR: 38  
NAMEF: S  
NAMET: S  
MESU : S  
EX :Z  
NOVA: Ø  
BACKS: NIL STREAM  
NEXVAR: 39  
VALUES : NILDAT

VAR: 39  
NAMEF: S

NAMET: S  
MESU : S  
EX :Z  
NOVA: 0  
BACKS: NIL STREAM  
NEXVAR: 40  
VALUES : NILDAT

VAR: 40  
NAMEF: S  
NAMET: S  
MESU : S  
EX :Z  
NOVA: 0  
BACKS: NIL STREAM  
NEXVAR: 41  
VALUES : NILDAT

VAR: 41  
NAMEF: S  
NAMET: S  
MESU : S  
EX :Z  
NOVA: 0  
BACKS: NIL STREAM  
NEXVAR: 42  
VALUES : NILDAT

VAR: 42  
NAMEF: S  
NAMET: S  
MESU : S  
EX :Z  
NOVA: 0  
BACKS: NIL STREAM  
NEXVAR: 43  
VALUES : NILDAT

VAR: 43  
NAMEF: S  
NAMET: S  
MESU : S  
EX :Z  
NOVA: 0  
BACKS: NIL STREAM  
NEXVAR: 44  
VALUES : NILDAT

VAR: 44  
NAMEF: S  
NAMET: S  
MESU : S  
EX :Z  
NOVA: 0  
BACKS: NIL STREAM  
NEXVAR: 45

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

MESU : S

EX : Z

NOVA: Ø

BACKS: NIL STREAM

NEXVAR: NIL NEXT VAR

VALUES : NIL DAT

