Department of Applied Analysis
and Computer Science
Technical Report CSTR 1003
July, 1970

THE DIALATOR SYSTEM
GENERAL PROGRAMMER'S GUIDE

by

Doron J. Cohen, Paul M. Fawcett
Eric G. Manning & Larry Smith

Faculty of Mathematics

University of Waterloo
Waterloo, Ontario
Canada

Department of Applied Analysis
and Computer Science
Technical Report CSTR 1003
July, 1970

THE DIALATOR SYSTEM
GENERAL PROGRAMMER'S GUIDE

by

Doron J. Cohen, Paul M. Fawcett
Eric G. Manning & Larry Smith

We wish to thank the Defence Research Board of Canada, the Northern Electric Research & Development Laboratories, Ottawa, and the Faculty of Mathematics of the University of Waterloo for financial, technical, and moral support which made this system of programs possible.

Other manuals in this series are:

1. TRAIZE User's Guide
2. FAUST User's Guide
3. TRAIZE Programmer's Guide
4. FAUST Programmer's Guide
5. File System User's Guide
6. File System Programmer's Guide

DIALATOR SYSTEM - GENERAL PROGRAMMER'S GUIDE

This is a programmer's overview for the DIALATOR system of programs. Its purpose is to allow the programmer to look at the 'forest' before getting too close to the trees. Thus it provides a panoramic overview of the whole DIALATOR system of programs.

It is recommended that SECTION 1) of this manual be read by a programmer wishing to study any part of the system. The guide is laid out as follows:

SECTION 1. Introduction

SECTION 2. Structure of the system

SECTION 3. PL1 Structures in the system

SECTION 4. Maintenance of the system

SECTION 5. Design Features of the DIALATOR

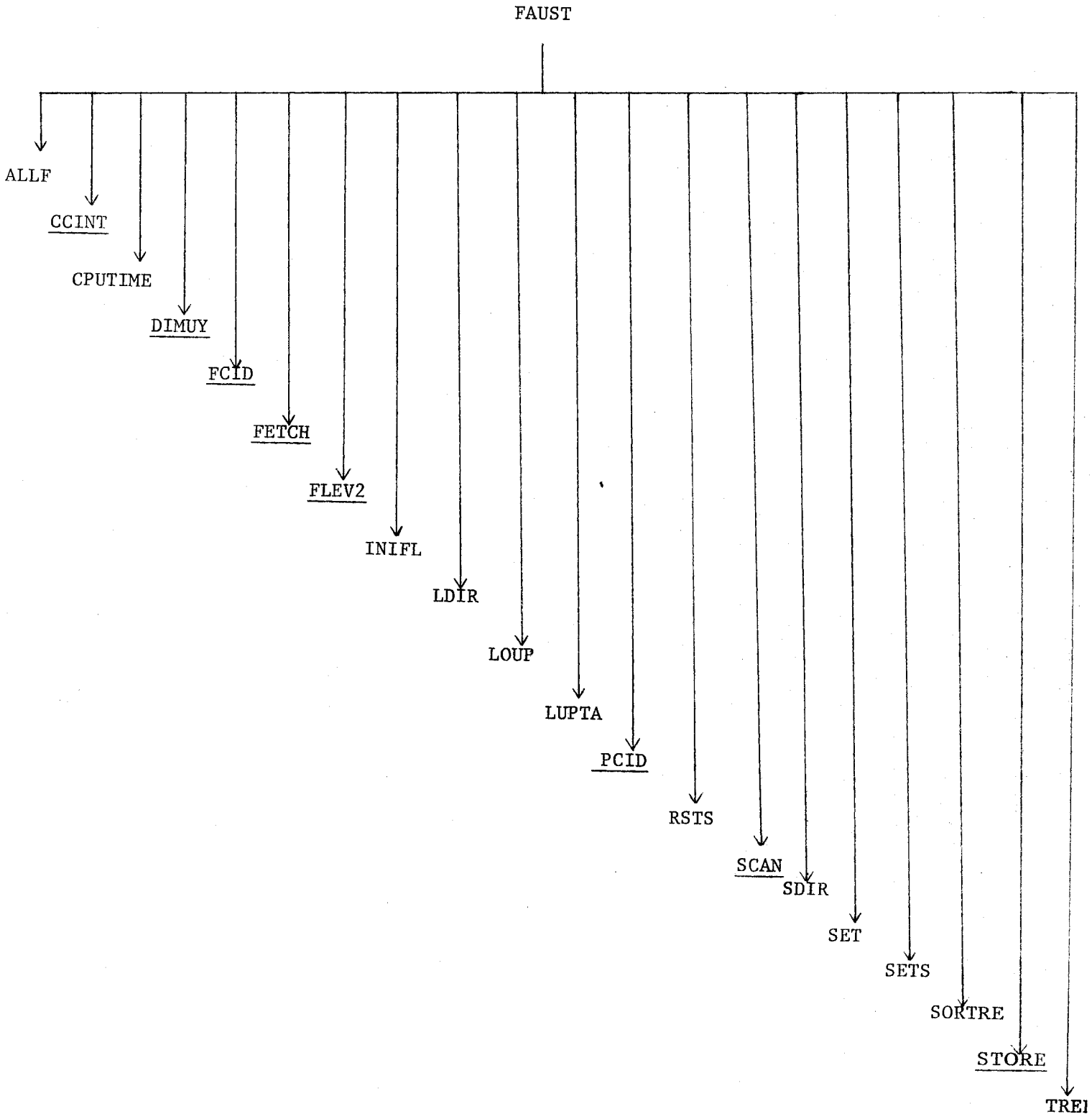
SECTION 1. Introduction.

This manual contains information necessary for the complete understanding of the system and the other programmer's guides.

Before the programmer tackles the programming of the system he should be familiar with the system as a user. SECTION 2 should be read before the other three programmer's guides are studied. The diagram of the program calling structure should be helpful during the study of the other programmer's guides. SECTION 3 should be well understood before starting any of the programming manuals. The remainder of this manual can be read either before or after the other guides have been studied.

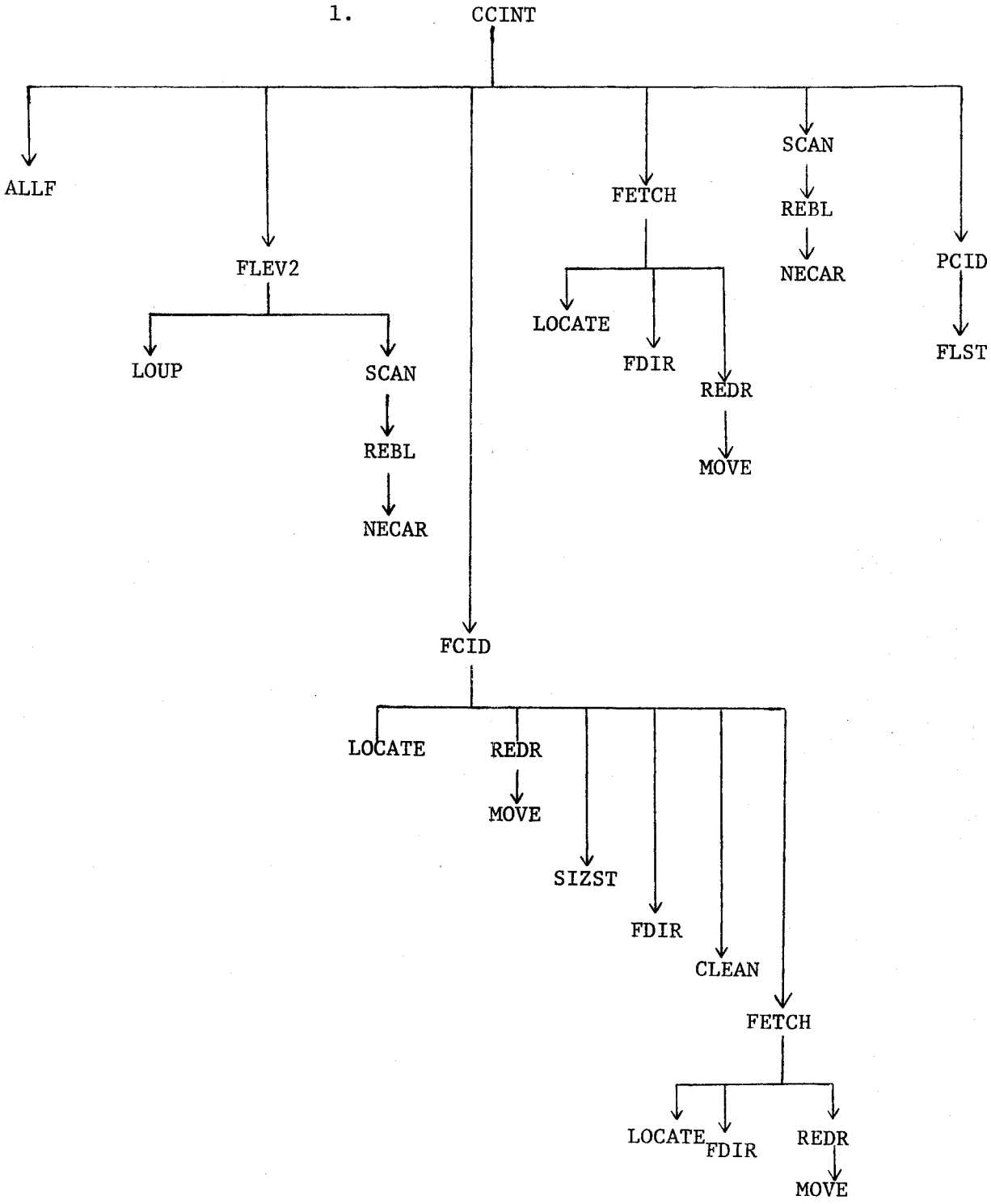
The FAUST and TRAIZE guides can be studied independently of the FILE SYSTEM guide, if the file subroutines called in FAUST and TRAIZE are viewed as 'magic' instructions. It may be more meaningful and, certainly will be closer to the user's world if the FILE guide is studied last.

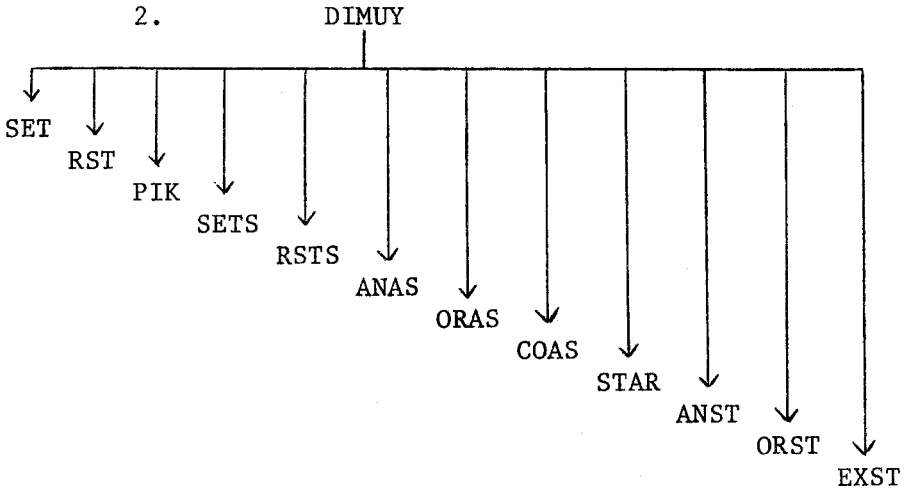
It is helpful to study the called routines in FAUST and TRAIZE guides before the main programs. In some cases it is necessary.

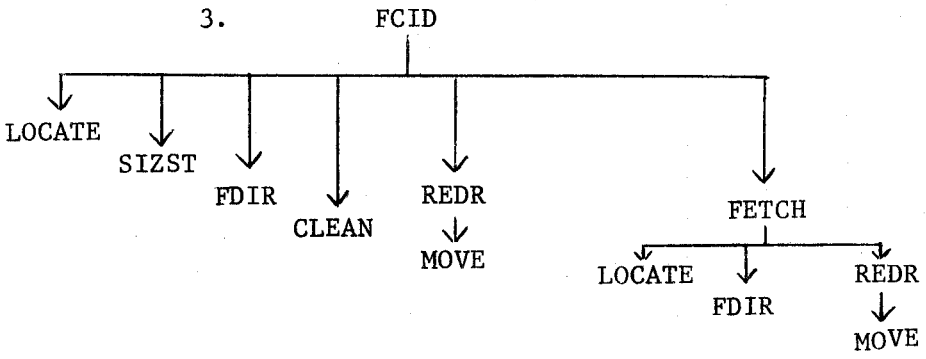


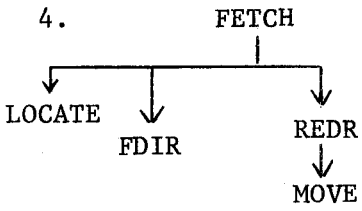
The underlined subroutines call others. Their subtrees are listed below.

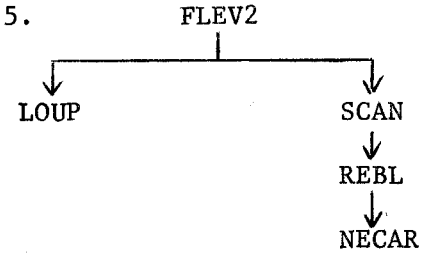
1.


CCINT
ALLF
FLEV2
LOUP
SCAN
REBL
NECAR
FETCH
LOCATE
FDIR
REDR
MOVE
SCAN
REBL
NECAR
PCID
FLST
FCID
LOCATE
REDR
MOVE
SIZST
FDIR
CLEAN
FETCH
LOCATE
FDIR
REDR
MOVE


2.
DIMUY
SET
RST
PIK
SETS
RSTS
ANAS
ORAS
COAS
STAR
ANST
ORST
EXST


3.
FCID
LOCATE
SIZST
FDIR
CLEAN
REDR
MOVE
FETCH
LOCATE
FDIR
REDR
MOVE


4.
FETCH
LOCATE
FDIR
REDR
MOVE


5.
FLEV2
LOUP
SCAN
REBL
NECAR

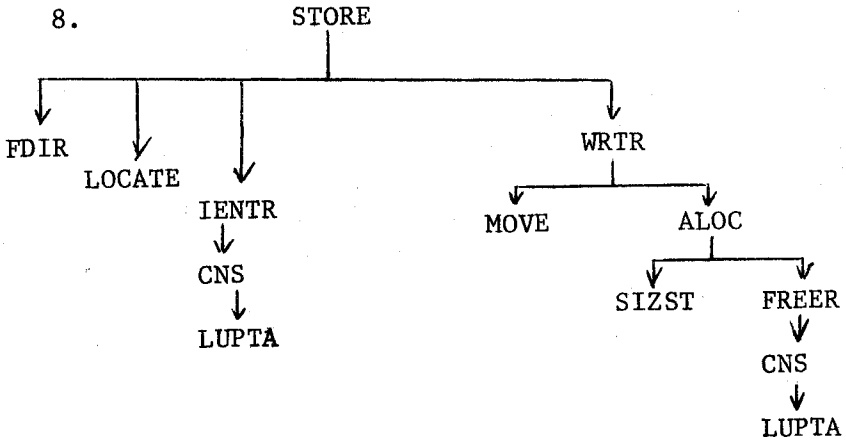
6.


PCID
FLST

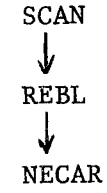
7.


PUTST
SIZST

8.


STORE
FDIR
LOCATE
IENTR
CNS
LUPTA
WRTR
MOVE
ALOC
SIZST
FREER
CNS
LUPTA

9.


SCAN
REBL
NECAR

SECTION 3. PL/1 Structures of the system

The PL/1 structures are of two types, those which contain information pertinent to single circuits, and those which contain general information. Each structure has a record name associated, which contains the name of the circuit if the structure is of the first type.

STRUCTURE 1 is $COPT.

This structure holds the circuit operator table and function catalogue with fault definitions. The information contained appears as follows:

CO# is the number of circuit operators.

CN# is the number of circuit functions.

CF# is the number of circuit functions.

COPTA is the circuit operator table (COPTA is shown below). The circuit operators, COPR, are the valid keywords and lead functions for TRAIZE. Along with each operator is given, an address, COAD. When an operator is looked for in the table the corresponding address is returned.

FCTLG is the functions catalogue. For each lead function it gives: #IN, the number of allowed input references, #FL, the maximum number of faults, FLT, a list of the faults, specifying the type. Notice, in the diagram of $COPT below, the input faults are listed last. The other faults are referred to as basic faults.

FLD is a vector describing each type of fault.

===== STRUCTURE NO. 1 =====

SIZE: 2610 BYTES, RECORD NAME:$COPT

$COPT.CO#= 22 $COPT.CN#= 14
$COPT.CF#= 4;

===== CIRCUIT OPERATORS TABLE =====

NO.	OPERATOR	ADDRESS
43	AND	13
44	AND2	3
45	END	64
46	FBK	2
47	FBKS	36
48	FF	37
49	FF0	10
50	FF1	9
51	INP	1
52	INPS	34
53	INPUTS	34
54	NAND	8
55	NAND2	6
56	NOR	12
57	NOT	7
58	OR	14
59	OR2	4
60	OUP	33
61	OUPS	35
62	OUTPUTS	35
63	WOR	11
64	XOR2	5

===== CIRCUIT FUNCTIONS CATALOGUE =====

NO.	FUNC.NAME	#IN	#FL	FAULTS TYPES							
1	INP	0	0	0	0	0	0	0	0	0	0
2	FBK	1	0	0	0	0	0	0	0	0	0
3	AND2	2	4	1	2	3	3	0	0	0	0
4	OR2	2	4	1	2	3	3	0	0	0	0
5	XOR2	2	4	1	2	3	3	0	0	0	0
6	NAND2	2	4	1	2	3	3	0	0	0	0
7	NOT	1	3	1	2	3	0	0	0	0	0
8	NAND	5	7	1	2	3	3	3	3	3	0
9	FF1	5	8	1	2	4	3	3	3	3	3
10	FF0	5	8	1	2	4	3	3	3	3	3
11	WOR	5	0	0	0	0	0	0	0	0	0
12	NOR	5	7	1	2	3	3	3	3	3	0
13	AND	5	7	1	2	3	3	3	3	3	0
14	OR	5	7	1	2	3	3	3	3	3	0

===== FAULT DESCRIPTION =====

1 OUTPUT STUCK AT ZERO
2 OUTPUT STUCK AT ONE
3 OPEN INPUT DIODE FROM
4 OPEN FEEDBACK DIODE

STRUCTURE 2 is $CIDS. This structure holds the circuit parameters. The information contained appears as follows:

N# is the number of leads in the circuit.

Ø# is the number of primary outputs in the circuit.

I# is the number of primary inputs in the circuit.

F# is the number of feedback loops in the circuit.

M# is the number of single faults in the circuit.

S# is the number of entries in the successors list. Note that this is not the number of successors since the pointers are included also.

T# is the number of terminals of the circuit.

E# is the number of lead names, which is not always equal to the number of leads.

R# is the number of simulation runs (this variable is not in use at the present time).

V# is the number of levels in the circuit.

Suppose we consider the following circuit description:

```
%RUN:       TRAIZE     CIRCUIT=CIRCUIT2,DISP=NO;
            INPUTS     (2)X1,X2;
            OUTPUTS    (1) Ø1;
  X1:       INP        /A1;
  X2:       INP        /A1,Ø1;
  A1:       AND        X1,X2/Ø1;
  Ø1:OUT:   OR         A1,X2;
            END        CIRCUIT2;
```

The values in $CIDS would be N#=4, Ø#=1, I#=2, F#=0, M#=9, S#=8, T#=0, E=5, R#=0, V#=3.

STRUCTURE 3 is $LDSC. This structure holds the leads description of the circuit. The information contained appears as follows:

N# is the number of leads in the circuit.

LEADS is a table with the lead function numbers, FNC, and the level, LVL, in which the lead has been placed. The ordering in which the lead information appears is the line ordering which has been assigned to the leads. This is not necessarily the ordering given by the user. Recall the mapping in TRAIZE. The input references, IR, are given by lead number. A pointer, OR, to the successor's list, in which the lead's output reference lead numbers are found, is given (See $SUCS).

For CIRCUIT2 described above $LDSC would be

N#=4, LEADS.FNC(1) = 1, LEADS.LVL(1)=0
LEADS.FNC(2) = 1, LEADS.LVL(2)=0
LEADS.FNC(3) = 13, LEADS.LVL(3)=1
LEADS.FNC(4) = 14, LEADS.LVL(4)=2
LEADS.IR(3,1)=1, LEADS.IR(3,2)=2,
LEADS.IR(4,1)=2, LEADS.IR(4,2)=3.

All other values of the IR array are zero.

LEADS.OR(1)=1,LEADS.OR(2)=3,
LEADS.OR(3)=6,LEADS.OR(4)=0.

STRUCTURE 4 is $FMCS. This structure has all the single faults for a specified circuit. The information appears as follows:

N# is the number of circuit leads.

FM is the faults table. Its dimensions are N# by eight. Eight is used because there is a maximum of eight single faults per lead.

To fill in the table each lead is examined and the associated faults are entered as numbers starting at one. The table, then associates a fault number with a lead and a type of fault. For CIRCUIT2 described above, FM would look like

	1	2	3	4	5	6	7	8
X1	0	0	0	0	0	0	0	0
X2	0	0	0	0	0	0	0	0
A1	1	2	3	4	0	0	0	0
01	5	6	7	8	0	0	0	0

STRUCTURE 5 is $SYMTA. This structure contains the lead names of the circuit. Since there may be more than one lead name per lead, back links are given for each lead name. The information contained appears as follows:

E# is the number of lead names in the circuit.

N# is the number of leads in the circuit.

TAB is the symbol table or lead name table, NAME. With each lead name, a lead number, LNM, is associated. This is the same lead number associated with the lead function of $LDSC.

BLINK is a set of back links so that, given a lead number, a lead name can be associated with it. (This lead name will be an arbitrary one of the multiple names for a given lead).

For CIRCUIT2 described above, we have E#=5, N#=4.

TAB.NAME(1)='A1', TAB.LNM(1)=3
TAB.NAME(2)='OUT', TAB.LNM(2)=4
TAB.NAME(3)='Ø1', TAB.LNM(3)=4
TAB.NAME(4)='X1', TAB.LNM(4)=1
TAB.NAME(5)='X2', TAB.LNM(5)=2

BLINK(1)=4, BLINK(2)=5, BLINK(3)=1, BLINK(4)=2

STRUCTURE 6 is $REFS. This structure contains input, output and feedback references. The information contained appears as follows:

I# is the number of inputs in the circuit.

O# is the number of outputs in the circuit.

F# is the number of feedbacks in the circuit.

IL is a list of lead numbers of the inputs.

OL is a list of lead numbers of the outputs.

FL is a list of lead numbers of the feedback nodes.

For CIRCUIT2, described above we have I#=2, O#=1, F#=0, IL(1)=1, IL(2)=2, OL(1)=4.

STRUCTURE 7 is $SUCS. This structure contains a list of successors for a given circuit. The information contained appears as follows:

S# is the number of entries in the successors list including pointers.

SS is the successors list.

For CIRCUIT2 described above we have outlined the structure $LDSC, STRUCTURE 3. For that structure the output reference pointers were OR(1)=1, OR(2)=3, OR(3)=6, OR(4)=0. These are necessary in the formation of $SUCS.

For CIRCUIT2, $SUCS would be:

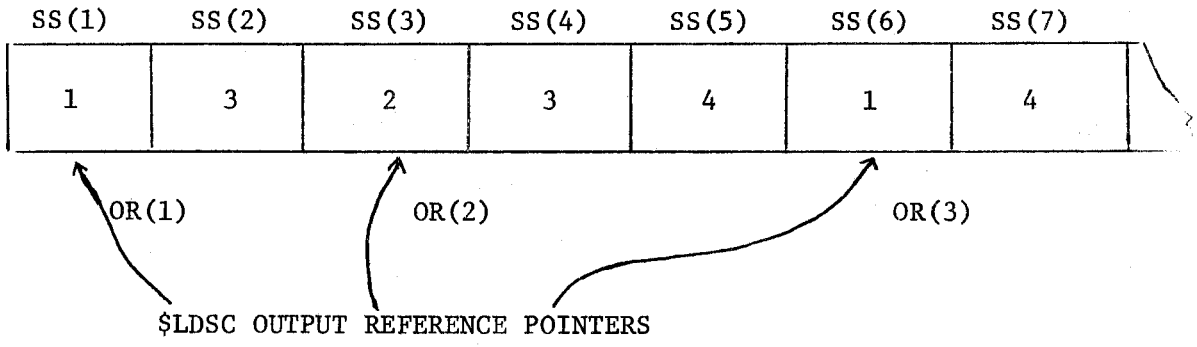
S#=8,

SS(1)=1, SS(2)=3

SS(3)=2, SS(4)=3, SS(5)=4

SS(6)=1, SS(7)=4,

Diagramatically the successors list looks as follows:



Pointer OR(I) points to the beginning of the list of successors for the Ith lead in $LDSC. The information contained in the elements of $SUCS pointed to, (above they are SS(1), SS(3), and SS(6)) is the number of elements of $SUCS which follow as successors of the Ith lead. Above, the first lead has one successor and its lead number is 3 which represents A1. The second lead has two successors and their lead numbers are 3 and 4.

STRUCTURE 8 is $TERM. This structure has not been implemented yet. It will involve terminals.

STRUCTURE 9 is $SFMC. This structure contains the selected faults for a given circuit. The information contained appears as follows:

N# is the number of leads of the circuit.

NF# is the number of faulty leads of the circuit.

M# is the number of faults selected.

FP is a vector of fault pointers. Each element of the vector is a mapping from the lead number to the row in the fault table with the faults of the lead. If a lead has no faults, its pointer is zero.

Suppose in CIRCUIT2 we define the selected faults as follows:

A1 s @ 1/X1;

O1 s @ 1, s @ 0;

$SFMC would be:

N#=4, NF#=2, M#=4,

FP(1)=0, FP(2)=0, FP(3)=1, FP(4)=2,

The $SFMC.FM table would look as follows:

	1	2	3	4	5	6	7	8
1	0	1	2	0	0	0	0	0
2	4	3	0	0	0	0	0	0

STRUCTURE 10 is $CTRES. This structure is yet to be defined. At the present time, if the user commands that the LVBS table and the diagnostic tree be saved, the recname of the structure saved will include $CTRES. There is no information associated with $CTRES.

STRUCTURE 11 is $DIATR. This structure contains the diagnostic tree of some circuit as a result of a FAUST run on the circuit. The information contained appears as follows:

M# is the number of faults in the circuit.

UB is the upper bound of the tree.

O# is the number of outputs of the circuit.

AV is a pointer to the first node in the available list of unused nodes.

FALIST is the fault option associated with the run.

BR is the brother links of the nodes of the tree.

SO is the son links of the nodes of the tree.

OU is the output vector associated with the nodes of the tree.

Notice the son and output vector are absent for the first M# nodes. The reason will become evident later. The output bit string is rounded to the nearest byte with function CEIL. The reason is that the assembler routines which handle the bit strings operate on strings which are byte multiples.

STRUCTURE 12 is $TEDS. This structure has not been implemented yet. It concerns the storing of test descriptions.

STRUCTURE 13 is $LVBS. This structure contains the LVBS table which gives the value of every circuit lead under every possible fault (which has been defined by the user). The information is as follows:

N# is the number of leads of the circuit.

M# is the number of faults.

LVBS is the table containing the lead values under all possible faults defined. The extra rows in the table are buffer rows. LVBS (-4) indicates an unstable machine with a '1' bit in the appropriate column. LVBS (-3), LVBS (-2), and LVBS (-1) are employed for the NAND flipflop only. LVBS(0) is a general buffer row used in boolean operations performed on the rows from 1 to N#.

@ is an activity vector with a bit for each lead. This vector enables the system to bypass leads that are not affected by a circuit stimulation. The activity bits can also be manipulated by the user. Each of the bit strings are rounded to the nearest byte to be compatible with the assembler routines which manipulate the bit strings.

STRUCTURE 14 is $OPTAB. This structure contains the general purpose operator table. The information is as follows:

OP# is the number of operators in the table.

TABLE is the operator table with a list of operators, OPER, and a list of addresses, ADDR, one to be associated with each operator.

In the DIALATOR system there are two $OPTAB's. One is TRAIZE.$OPTAB, which holds all the keywords of the traize language and the valid lead function names. The other is FAUST.$OPTAB, which holds all the instruction words in the FAUST language.

Of the above 14 structures, numbers 10 through 13 will not be understood fully until the FAUST Programmer's Guide is studied.

In addition to the 14 structures there are two structures CARDIN and JOBPARM which are discussed below.

CARDIN is stored in SYSLIB under the name SD. This structure contains variables necessary to read source cards for TRAIZE or FAUST descriptions. The information contained is as follows:

SOCAR is a vector of length 72 containing the characters from the first 72 columns of a card. The last 8 columns are ignored.

LINM is the line number of the card and is picked up from the last 8 columns of the card.

DELIS is a string of valid delimiters for the TRAIZE and FAUST languages. It is initialized in SUBROUTINE CCINT.

QUT, QUP are the characters to be recognized as valid opening and closing parenthesis respectively. Any collection of characters found between these parentheses will be bypassed as comments.

EOD is a label switch indicating the action to be taken if the data found dictates special action.

EOF is a label switch indicating the action to be taken if the data cards terminate prematurely. Also action for failure of the system is contained in EOF.

SOUP is a pointer to the column of the card being examined at present.

LEN holds the length of the current syntactic unit found in NSU.

NC points to the position of a delimiter in the string DELIS. If the delimiter is not present in the string DELIS, NC is zero.

JOBPARM is stored in SYSLIB under the name JOBPARM. This structure contains the information given in the TRAIZE or FAUST control card. The information contained is as follows:

CCC, STATUS, COMCODE, NPARM1, NPARM2, NPARM3 are not implemented yet.

RUNAME is the runname, picked up as the first syntactic unit after the % sign.

OPR is the operation name. It will be 'TRAIZE' if the control card is that of a circuit description. It will be 'SIMULATE' if the control card is that of a FAUST run.

FLOP is the disposition of the run results. FLOP can have values NO, YES, OLD or NEW. NO indicates the results are not to be stored in the file system. YES indicates the results are to be stored in the file system. OLD indicates that if a file with the same file name is found it is to be rewritten. NEW indicates that if a file with the same file name is not found the results are to be stored.

CNAME is the circuit name. This is the name used as part of the file name when results are stored in the file system.

SECTION 4. Maintenance of the system.

The system consists of six partitioned data sets stored on disc. Of these, two are load modules.

They are named as follows:

SLIB1 is a source library of external PL/1 subroutines.

LLIB1 is a load module library of the programs from SLIB1.

SMAN2 is a source library of PL/1 mainline programs.

LMAN2 is a load module library of programs from SMAN2.

SYSUT is a source library of utility programs for the system.

SYSLIB is a source library of DCL statements for the external subroutines and PL/1 structures to be included at compile time (via the PL/1. % INCLUDE feature), plus three overlay structures for the main programs.

Programs of SYSUT:

1. SYSUT has a program which copies the partitioned data sets onto tapes. There are three data sets included in this program which are not available. They are SLIB2, SLIB3 and LLIB2. If they are included in any of the SYSUT programs they should be deleted. Their purpose was to aid the development of the DIALATOR system only. To run submit source as an OS job.

2. There are clean up programs provided, which condense the data sets on the disc. They are:

CUPALL to clean up all data sets,
CUPLL1 to clean up LLIB1,
CUPLM2 to clean up LMAN2,
CUPSL1 to clean up SLIB1,
CUPSM2 to clean up SMAN2,
CUPSYS to clean up SYSLIB.

3. There is a program, DELNODE, which deletes a node and all its subset nodes from the tree. The node name must be specified to the function DELETE. (To use refer to the File System User's Guide.)

4. There is a program which is called DIALTR which does a combined TRAIZE - FAUST run. This program enables the user to describe and simulate a circuit without first storing the circuit description in the file system. To use see RUNDIAL of this section.

5. INICOPT is a program which will initialize $COPT. The old $COPT must be deleted from the file system before the new one is entered. This program is valuable if the present $COPT becomes obsolete in any respect. To change fault definitions etc. consult the INICOPT program source statements and the copy of $COPT in section 3) of this manual.

6. LISTDIR is a program which executes the load module of LISTDIR on LMAN2. This program lists the file directory. (To use this program see the File System User's Guide.)

7. MEMBERS lists the members of each partitioned data set. To run this program, submit the source as an OS job (Through WITS type 'RUN SYSUT (MEMBERS)').

8. PRINTEM prints all source libraries. To run submit source as an OS job (Through WITS type 'RUN SYSUT(PRINTEM)).

9. There is a program which will print or punch any specified member. It is called PUNCH. For printing put

SYSUT2 DD SYSOUT=A,

for punching put

SYSUT2 DD SYSOUT=B.

To run the program submit as an OS job.

10. RUNDIAL is a program which will execute the load module form of DIALTR on LMAN2. To run place your circuit description followed by your faust statements after the card

//GO.SYSIN DD *

and submit this modified source as an OS job.

11. RUNFAUST is a program which will execute the load module of FAUST on LMAN2. To run this program consult the FAUST User's Guide.

12. TRARUN is a program which executes the load module of TRAIZE stored on LMAN2. To run this program consult the TRAIZE User's Guide.

13. SETA is a program which will list, create or modify the structure $OPTAB.

Description of variables:

TNAME is the name of the operator table.

OPR is the operator to be placed in, or delete from, the table.

ACT is the act desired by the user. It can have the value CREATE, MODIFY, or LIST depending on whether the user wishes to create, modify or list $OPTAB. Also with respect to each operator in the data ACT can have the value ALTER, REMOVE or END. ALTER indicates the

operator is to be changed. REMOVE indicates the operator is to be removed. END must always appear before the last operator in the data to signal the end of the data list.

DISP is the disposition associated with the data. It can be YES, NO, NEW, or OLD.

TIT is the title associated with the structure.

SIZE is the size of the structure.

ADR is the address associated with the operator name.

L is the line number.

The included subroutines are ENTA, FETCH, STORE, FDIR, SIR, and LDIR.

MOR:

The parameters are set to default values. TNAME, DISP, ACT and SIZE are read from the data. They are checked for conflicts. If the act is LIST go to LIST. If TNAME is blank go to FIN.

GLOOP:

The operators, their addresses and the associated act are read while ACT is not 'END'. If ACT is not any of REMOVE, ALTER or END, it is assumed that the operator is to be added to the table. If ACT is ALTER and the operator has been found it is altered. If it is not found a message is printed. If ACT is REMOVE and the operator has been found it is removed. If it is not found a message is printed. If the act is none of these and the operator is found a message is printed. If it is not found ENTA is called to enter it into the table.

If the DISP is appropriate the resulting table is stored.

LIST:

The table is listed. A transfer is done to MOR, to pick up more information.

FIN:

The directory is printed.

In describing the data care must be taken to conform to the format statements. Also each time the parameters TNAME etc. are read. A card must follow (even if blank to pick up TIT).

All programs described above are set up to run a system /360 computer under OS/360. JCL must be changed for your specific installation if it is not compatible.

SECTION 5. Design Features of the DIALATOR.

The DIALATOR system consists of a collection of PL/1 and 360 Assembler routines. Most of the routines are small PL/1 external subroutines. By writing the system this way maintenance is much easier. Debugging can be done quickly on the individual programs and replacement by 360 Assembly coding or modification of subroutines is a simpler task.

Execution time may suffer in some cases because of the many subroutines. This is increased by overlaying, due to the manipulation of subroutines during execution time. However, the greatly reduced core requirement in using overlays is sufficient justification for their use. Both FAUST and DIALTR have overlay structures.

The system was coded in PL/1 because of the ease in coding and the many features available in the PL/1 language.

The system has its own file system which is custom-tailored to suit the DIALATOR needs. Thus the system is more independent than it would be otherwise.