Department of Applied Analysis
and Computer Science
Research Report CRGO 2817
March, 1970

THE DIALATOR SYSTEM
TRAIZE USER'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.
This is a user's guide to program TRAIZE of the DIALATOR system. (A system programmer's guide, explaining internal workings of the program, appears later). This guide is laid out as follows:

**Section 1** - Tells how to describe a logic circuit to the program.

**Section 2** - Tells how to punch the circuit description on cards.

**Section 3** - Describes the Job Control Language cards needed to make a TRAIZE run on a System/360 computer under OS/360.

**Section 4** - Describes the output and what it means.

**Section 5** - Contains TRAIZE run examples.

**Section 6** - Error Diagnostics.

---

**Section 1**

**How To Describe A Circuit**

Here is a circuit. Let us refer to it from now on as CIRCUIT1.

![Diagram 1](image-url)
Here is how we describe CIRCUIT1 to TRAIZE:

(column 1 of card)

% RUN: TRAIZE CIRCUIT=CIRCUIT1, DISP=NO;
INPUTS (2) X1,X2;
OUTPUTS (2) Ø1,A2;

X1: INP;
X2: INP;
A1: AND X1,X2;
N1: NAND X1,X2;
A2: AND X1,A1;
Ø1: ØR A1,N1;
END CIRCUIT1;

DIAGRAM 2

In CIRCUIT1 (see diagram 1) there are six points with **labels**. These labelled points are called the **leads** of CIRCUIT1.

Each lead performs a **function** in CIRCUIT1. X1 and X2 are primary inputs. A1 and A2 are AND gates. N1 is a NAND gate. Ø1 is an ØR gate. Also A2 and Ø1 are outputs for CIRCUIT1.

To describe CIRCUIT1 in words, for each lead we could give (see diagram 2 - lines 4 to 9):

1) its label.
2) the function (NAND, NOR, INPUT, etc.) it performs.
3) the labels of the leads which feed it (its inputs) or we could give the labels of its outputs.
In TRAIZE, we must give the input labels. We will see later how output labels can be optionally supplied to provide a powerful check for user coding errors.

Section 2

Circuit Description On Computer Cards

Order And Content Of Card Statements

To make it easy for the system to read the description of CIRCUIT1 a certain order is followed (see diagram 2)

1. Control card which contains
   a) a run name ----- RUN
   b) the keyword TRAIZE
   c) a circuit name ----- CIRCUIT1
   d) a disposition code ----- NO

2. Statement which contains
   a) keyword ----- INPUTS
   b) number of inputs ----- 2
   c) list of inputs ----- X1, X2

3. Statement which contains
   a) keyword ----- OUTPUTS
   b) number of outputs ----- 2
   c) list of outputs ----- A2, Ø1

4. Statement which contains
   a) label of lead ----- X1
   b) function of lead ----- INF
   c) input references of lead ----- none
5. Additional statements as in 4.

6. Statement which contains
   a) keyword --- END
   b) a circuit name --- CIRCUIT

II Card Format Rules

A) Control Card (see diagram 2 - line 1)
   1. "$%" must appear in column 1
   2. followed by run name or program name
   3. followed by ":"
   4. followed by "TRAIZE"
   5. followed by at least one blank
   6. followed by "CIRCUIT"
   7. followed by "="
   8. followed by the circuit name
   9. followed by ","
   10. followed by "DISP"
   11. followed by "="
   12. followed by "NO" *
   13. followed by ";"

   The control card must be the first card in the deck (except for JCL cards). The "$%" sign must be used only in the control card.

* Later we will see other values for DISP.
B) **Header Statement Cards** (see diagram 2, lines 2 & 3)

1. First word on the card must be one of the following keywords: "INPUTS"
   "OUTPUTS"
   "FEEDBACKS"

2. followed by "("  
3. followed by number of inputs, outputs, or feedbacks  
4. followed by ")"  
5. followed by at least one blank  
6. followed by an input, output or feedback label  
7. followed by (a) "," if more labels  
   (b) ";" if last label  

Inputs and outputs statements must appear; feedbacks statement only if there are feedback loops in the circuit. The header statement cards must follow the control card; their relative order does not matter, however.

i.e. outputs statement can appear before inputs statement.

C) **Lead Statement Cards** (see diagram 2, lines 4 to 9)

1. First word is lead label  
2. followed by ":"  
3. followed by lead function  
4. followed by at least one blank  
5. followed by an input reference label  
6. followed by a) "," if more input references  
   b) ";" if last input reference
The lead statements must appear after the header statements; their relative order does not matter, however.

D) **End Statement**

1. First word is keyword "END"
2. followed by at least one blank
3. followed by circuit name
4. followed by ";"

III **Other Rules**

1. The card description is totally free format and blank is a delimiter. Thus any number of blanks may appear between fields and a field may start in any column. Moreover, a statement can overflow into one or more cards with no special action.

2. No word or number may have intervening blanks.
   
   eg. in CIRCUIT1 Card l 6 6 TRÄIZE is invalid.

3. Comments may be inserted between any fields and must be enclosed in quotes.
   
   eg. Xl: INP; "Xl IS AN INPUT"

4. Columns 73 through 80 are ignored.

5. Lead labels may not exceed 6 characters in length.

6. Run name and Circuit name may not exceed 8 characters in length.
Option

The programmer may list the output references, for any lead, in the lead statement. For example, in CIRCUIT1 the output reference labels for lead A1 are \( \emptyset 1 \) and A2. The lead statement for A1 with output references would read:

\[
A1: \quad \text{AND} \quad X1,X2/\emptyset 1,A2;
\]

RULES

1) programmer must list all or none of the output references for a particular lead but need not do output references for all leads.

2) input references and output references must be separated by a "/".

**NOTE:** If you give the optional output references for every lead of a circuit, program TRAIZE will make a very thorough check of your coding work for you. This is possible because either the input references or the output references alone provide enough information to completely define your circuit. Hence if you give both, you have really described your circuit twice. TRAIZE takes advantage of this by comparing the two descriptions and complaining loudly if any discrepancies are found.

The practice of giving both input and output references is therefore highly recommended, whenever the simulation results must be trustworthy.
Section 3) Control Cards for TRAIZE

The following deck is an example of the necessary control cards used to make a TRAIZE run on a system 360 under OS/360.

//TRAIZE JOB 'U0664NORELEC,TIME=(,20),PAGES=99',MSGLEVEL=1,
   //      REGION=250K
//JOBLIB DD DSN=P0785.LMAN2,DISP=SHR,UNIT=2314
//      EXEC PGM=TRAIZE,REGION=250K
//SYSPRINT DD SYSOUT=A
//RECFILE DD DSN=U0664.RECFILE,DISP=OLD,UNIT=2314
//RECDIR DD DSN=U0664.RECDIR,DISP=OLD,UNIT=2314
//SCSYMTA DD DSN=&SCSYMTA,DISP=(NEW,PASS),SPACE=(TRK,(10)),
   //      UNIT=2314,DCB=(RECFM=FB,BLKSIZE=200,LRECL=20)
//SCLDSC DD DSN=&SCLDSC,DISP=(NEW,PASS),SPACE=(TRK,(10)),
   //      UNIT=2314,DCB=(RECFM=FB,BLKSIZE=200,LRECL=40)
//SCSS DD DSN=&SCSS,DISP=(NEW,PASS),SPACE=(TRK,(10)),
   //      UNIT=2314,DCB=(RECFM=FB,BLKSIZE=200,LRECL=20)
//STRMIN DD DSN=&STRMIN,DISP=(NEW,PASS),SPACE=(TRK,(10)),
   //      UNIT=2314,DCB=(RECFM=FB,BLKSIZE=500,LRECL=100)
//SYSIN DD *

Your TRAIZE deck is inserted here.

/*
Section 4  
Output Description  
The first output is the control card of your description. Next will be the Operators Table Listing.

It will look like

```
= = = = OPERATORS TABLE LISTING = = = =

<table>
<thead>
<tr>
<th>NO.</th>
<th>SYMBOL</th>
<th>ADDRESS</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td></td>
<td>0</td>
</tr>
<tr>
<td>2</td>
<td>AND</td>
<td>13</td>
</tr>
<tr>
<td>3</td>
<td>END</td>
<td>64</td>
</tr>
<tr>
<td>4</td>
<td>FBK</td>
<td>2</td>
</tr>
<tr>
<td>5</td>
<td>FBKS</td>
<td>36</td>
</tr>
<tr>
<td>6</td>
<td>FEEDBACKS</td>
<td>36</td>
</tr>
<tr>
<td>7</td>
<td>INP</td>
<td>1</td>
</tr>
<tr>
<td>8</td>
<td>INPS</td>
<td>34</td>
</tr>
<tr>
<td>9</td>
<td>INPUTS</td>
<td>34</td>
</tr>
<tr>
<td>10</td>
<td>NAND</td>
<td>8</td>
</tr>
<tr>
<td>11</td>
<td>NOR</td>
<td>12</td>
</tr>
<tr>
<td>12</td>
<td>NOT</td>
<td>7</td>
</tr>
<tr>
<td>13</td>
<td>OR</td>
<td>14</td>
</tr>
<tr>
<td>14</td>
<td>OUP</td>
<td>33</td>
</tr>
<tr>
<td>15</td>
<td>OUPS</td>
<td>35</td>
</tr>
<tr>
<td>16</td>
<td>OUTPUTS</td>
<td>35</td>
</tr>
</tbody>
</table>
```
The symbols are all the possible keywords that a programmer can use. They consist of Header Statement and Lead Statement keywords, plus "END".

Optional Header Statement keywords are also listed; They are:

INPS for INPUTS
OUPS for OUTPUTS
FBKS for FEEDBACKS

This is followed by a listing of your source deck; that is, the program describing the circuit.

Next is a printout of the variables:

<table>
<thead>
<tr>
<th>Variable</th>
<th>Value in CIRCUIT1</th>
</tr>
</thead>
<tbody>
<tr>
<td>N# - number of leads in circuit</td>
<td>N# = 6</td>
</tr>
<tr>
<td>E# - number of labels in circuit</td>
<td>E# = 6</td>
</tr>
<tr>
<td>K# - pointer to next available space in $SUCS,SS</td>
<td>K# = 1</td>
</tr>
<tr>
<td>I# - number of inputs</td>
<td>I# = 2</td>
</tr>
<tr>
<td>O# - number of outputs</td>
<td>O# = 2</td>
</tr>
<tr>
<td>F# - number of feedbacks</td>
<td>F# = 0</td>
</tr>
<tr>
<td>JOBPARM,CNAME - circuit name</td>
<td>JOBPARM,CNAME = 'CIRCUIT1'</td>
</tr>
<tr>
<td>JOBPARM,RUNAME - run name</td>
<td>JOBPARM,RUNAME = 'RUN'</td>
</tr>
<tr>
<td>JOBPARM,DISP - disposition parameter</td>
<td>JOBPARM,DISP = 'NO'</td>
</tr>
</tbody>
</table>

The OPERATORS TABLE LISTING is printed out again as above.

Next the number of levels of the circuit is printed.

* K# explained in Programmer's Guide
Here are the levels in CIRCUIT1

---

**DIAGRAM 3**

By dividing CIRCUIT1 into levels, we can picture CIRCUIT1 as the union of 3 circuits:

- CIRCUIT1. LEVEL 0
- CIRCUIT1. LEVEL 1
- CIRCUIT1. LEVEL 2

The outputs of LEVEL 0 are the inputs of LEVEL 1 and the outputs of LEVEL 1 are the inputs of LEVEL 2.

The last output is a mapping of the programmer's ordering of the leads to required ordering of the leads.

In CIRCUIT1 the leads of LEVEL 0 must be first, followed by the leads of LEVEL 1, followed by the leads of LEVEL 2.

For example (see diagram 3)
(1)

<table>
<thead>
<tr>
<th>Lead</th>
<th>Programmer's Order</th>
<th>Required Order</th>
<th>Mapping Output</th>
</tr>
</thead>
<tbody>
<tr>
<td>X1</td>
<td>1</td>
<td>1</td>
<td>1 ---- 1</td>
</tr>
<tr>
<td>X2</td>
<td>2</td>
<td>2</td>
<td>2 ---- 2</td>
</tr>
<tr>
<td>A1</td>
<td>3</td>
<td>3</td>
<td>3 ---- 3</td>
</tr>
<tr>
<td>N1</td>
<td>4</td>
<td>4</td>
<td>4 ---- 4</td>
</tr>
<tr>
<td>A2</td>
<td>5</td>
<td>5</td>
<td>5 ---- 5</td>
</tr>
<tr>
<td>Ø1</td>
<td>6</td>
<td>6</td>
<td>6 ---- 6</td>
</tr>
</tbody>
</table>

If we had written the leads in a different order, the picture would change:

(2)

<table>
<thead>
<tr>
<th>Lead</th>
<th>Programmer's Order</th>
<th>Required Order</th>
<th>Mapping Output</th>
</tr>
</thead>
<tbody>
<tr>
<td>X1</td>
<td>1</td>
<td>1</td>
<td>1 ---- 1</td>
</tr>
<tr>
<td>Ø1</td>
<td>2</td>
<td>5</td>
<td>2 ---- 5</td>
</tr>
<tr>
<td>N1</td>
<td>3</td>
<td>3</td>
<td>3 ---- 3</td>
</tr>
<tr>
<td>A2</td>
<td>4</td>
<td>6</td>
<td>4 ---- 6</td>
</tr>
<tr>
<td>A1</td>
<td>5</td>
<td>4</td>
<td>5 ---- 4</td>
</tr>
<tr>
<td>X2</td>
<td>6</td>
<td>2</td>
<td>6 ---- 2</td>
</tr>
</tbody>
</table>

In general, to keep output simple try to list leads of LEVEL 0, followed by leads of LEVEL 1, etc., (in program deck).
Section 5

The first example is CIRCUIT1. The DISP used is YES. (An explanation of DISP values is contained in APPENDIX A of this manual.)

In addition to the normal output, described earlier, a listing of the file directory tree is printed. The reason is, we have asked to have the description of CIRCUIT1 stored on the file. If it has been stored, CIRCUIT1 should appear as a node in the tree.

Following this the file description of CIRCUIT1 is fetched and printed. If you understand this description you should be able to draw from it, CIRCUIT1, as shown in DIAGRAM 1, Section 1.

Here is the run output.
EXAMPLE 1

**RUN:** TRAIZE CIRCUIT=CIRCUIT1,DISP=YES;

**FDIR==OLD DIRECTORY FETCHED.**

**FETCH==TRAIZE.$OPTAB**

**STRUCTURE:** $OPTAB OF NODE NO. 51, FETCHED 228 BYTES, STARTING REGION NO. 11 WORD NO. 621

====== STRUCTURE NO. 14 =====

**SIZE:** 228 BYTES, RECORD NAME: TRAIZE.$OPTAB

$OPTAB.OP#= 16;

====== OPERATORS TABLE LISTING=====

<table>
<thead>
<tr>
<th>NO.</th>
<th>SYMBOL</th>
<th>ADDRESS</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td></td>
<td>0</td>
</tr>
<tr>
<td>2</td>
<td>AND</td>
<td>13</td>
</tr>
<tr>
<td>3</td>
<td>END</td>
<td>64</td>
</tr>
<tr>
<td>4</td>
<td>FBK</td>
<td>2</td>
</tr>
<tr>
<td>5</td>
<td>FBKS</td>
<td>36</td>
</tr>
<tr>
<td>6</td>
<td>FEEDBACK</td>
<td>36</td>
</tr>
<tr>
<td>7</td>
<td>INP</td>
<td>1</td>
</tr>
<tr>
<td>8</td>
<td>INPS</td>
<td>34</td>
</tr>
<tr>
<td>9</td>
<td>INPUTS</td>
<td>34</td>
</tr>
<tr>
<td>10</td>
<td>NAND</td>
<td>8</td>
</tr>
<tr>
<td>11</td>
<td>NOR</td>
<td>12</td>
</tr>
<tr>
<td>12</td>
<td>NOT</td>
<td>7</td>
</tr>
<tr>
<td>13</td>
<td>OR</td>
<td>14</td>
</tr>
<tr>
<td>14</td>
<td>OUP</td>
<td>33</td>
</tr>
<tr>
<td>15</td>
<td>OUPS</td>
<td>35</td>
</tr>
<tr>
<td>16</td>
<td>OUTPUTS</td>
<td>35</td>
</tr>
</tbody>
</table>
INPUTS (2) X1,X2;
OUTPUTS (2) O1,A2;
X1: INP;

***TRAIZE*** COMBINATIONAL CIRCUIT.
X2: INP;
A1: AND X1,X2;
N1: NAND X1,X2;
A2: AND X1,A1;
O1: OR A1,N1;
END CIRCUIT1;

( a second listing of TRAIZE.$OPTAB has been omitted )

PHASE BUILDER
THE GIVEN CIRCUIT PARAMETERS ARE

N#=       6    E#=      6    K#=       1    I#=       2
F#=       0    CNAME='CIRCUIT1'    J0BPARM.RUNAME='RUN'

==FDIR==OLD DIRECTORY FETCHED.
==FETCH==TRAIZE.$OPTAB
STRUCTURE: $OPTAB OF NODE NO. 51, FETCHED 228 BYTES, STARTING REGION NO. 11 WORD NO. 621
CIRCUIT1.$CIDS.$SUICS

CREATED.

==LECR==THE CIRCUIT ORGANIZED AND LEVELED IN: 3 LEVELS.

THE MAPPING:

1-> 1
2-> 2
3-> 3
4-> 4
5-> 5
6-> 6

==STORE==CIRCUIT1.$CIDS
STRUCTURE: $CIDS OF NODE NO. 71, STORED 86 BYTES, STARTING REGION NO. 12 WORD NO. 52

==STORE==CIRCUIT1.$CIDS.$LDSC
STRUCTURE: $LDSC OF NODE NO. 78, STORED 168 BYTES, STARTING REGION NO. 12 WORD NO. 547

==STORE==CIRCUIT1.$CIDS.$SYMTA
STRUCTURE: $SYMTA OF NODE NO. 81, STORED 124 BYTES, STARTING REGION NO. 12 WORD NO. 492

==STORE==CIRCUIT1.$CIDS.$REFS
STRUCTURE: $REFS OF NODE NO. 82, STORED 56 BYTES, STARTING REGION NO. 12 WORD NO. 589

==STORE==CIRCUIT1.$CIDS.$SUICS
STRUCTURE: $SUICS OF NODE NO. 83, STORED 75 BYTES, STARTING REGION NO. 0 WORD NO. 20

0# = 2
==== DIRECTORY LISTING ====

98 AVAIL POINTER.
40 REGIONS USED.
24 STRUCTURES RECOGNIZED.
136 NODES IN THE DIRECTORY.
141 DESIRED NO. OF NODES.
22 KEY-WORDS RECOGNIZED.
700308 DATE OF LAST CHANGE.

==== DIALATOR KEY-WORDS TABLE ====

<table>
<thead>
<tr>
<th>NO.</th>
<th>KEY-WORD</th>
<th>INDEX</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td></td>
<td>0</td>
</tr>
<tr>
<td>2</td>
<td></td>
<td>0</td>
</tr>
<tr>
<td>3</td>
<td></td>
<td>0</td>
</tr>
<tr>
<td>4</td>
<td></td>
<td>0</td>
</tr>
<tr>
<td>5</td>
<td></td>
<td>0</td>
</tr>
<tr>
<td>6</td>
<td></td>
<td>0</td>
</tr>
<tr>
<td>7</td>
<td></td>
<td>0</td>
</tr>
<tr>
<td>8</td>
<td></td>
<td>0</td>
</tr>
<tr>
<td>9</td>
<td>$GIDS</td>
<td>2</td>
</tr>
<tr>
<td>10</td>
<td>$COPT</td>
<td>1</td>
</tr>
<tr>
<td>11</td>
<td>$CTRES</td>
<td>10</td>
</tr>
<tr>
<td>12</td>
<td>$DIATR</td>
<td>11</td>
</tr>
<tr>
<td>13</td>
<td>$FMCS</td>
<td>4</td>
</tr>
<tr>
<td>14</td>
<td>$LDSC</td>
<td>3</td>
</tr>
<tr>
<td>15</td>
<td>$LVBS</td>
<td>13</td>
</tr>
<tr>
<td>16</td>
<td>$OPTAB</td>
<td>14</td>
</tr>
<tr>
<td>17</td>
<td>$REFS</td>
<td>6</td>
</tr>
<tr>
<td>18</td>
<td>$SFMC</td>
<td>9</td>
</tr>
<tr>
<td>19</td>
<td>$SUCS</td>
<td>7</td>
</tr>
<tr>
<td>20</td>
<td>$SYMTA</td>
<td>5</td>
</tr>
<tr>
<td>21</td>
<td>$TEDS</td>
<td>12</td>
</tr>
<tr>
<td>22</td>
<td>$TERM</td>
<td>8</td>
</tr>
</tbody>
</table>
#### THE DIRECTORY TREE ####

```
<table>
<thead>
<tr>
<th>No.</th>
<th>REC_NAME</th>
<th>SIZE</th>
</tr>
</thead>
<tbody>
<tr>
<td>STB</td>
<td>SRG</td>
<td>TYP</td>
</tr>
<tr>
<td>SIS</td>
<td>DAU</td>
<td></td>
</tr>
<tr>
<td>0</td>
<td>REC_FILE</td>
<td>0</td>
</tr>
<tr>
<td>-1</td>
<td>-1</td>
<td>4</td>
</tr>
<tr>
<td>1</td>
<td>HEADFREE</td>
<td>0</td>
</tr>
<tr>
<td>-1</td>
<td>-1</td>
<td>88</td>
</tr>
<tr>
<td>88</td>
<td>FREE</td>
<td>273</td>
</tr>
<tr>
<td>718</td>
<td>12</td>
<td>0</td>
</tr>
<tr>
<td>87</td>
<td>FREE</td>
<td>74</td>
</tr>
<tr>
<td>612</td>
<td>12</td>
<td>0</td>
</tr>
<tr>
<td>113</td>
<td>FREE</td>
<td>89</td>
</tr>
<tr>
<td>218</td>
<td>12</td>
<td>0</td>
</tr>
<tr>
<td>77</td>
<td>FREE</td>
<td>3</td>
</tr>
<tr>
<td>699</td>
<td>12</td>
<td>0</td>
</tr>
<tr>
<td>85</td>
<td>FREE</td>
<td>74</td>
</tr>
<tr>
<td>882</td>
<td>11</td>
<td>0</td>
</tr>
<tr>
<td>80</td>
<td>FREE</td>
<td>37</td>
</tr>
<tr>
<td>603</td>
<td>12</td>
<td>0</td>
</tr>
<tr>
<td>70</td>
<td>FREE</td>
<td>11</td>
</tr>
</tbody>
</table>
```

( Part of the tree print out has been omitted.

Section containing CIRCUIT1 & DECODER are shown on next page. )
***************

98 NODES IN THE DIR TREE  29 NODES IN THE FREE LIST.
89056 BYTES USED IN RECORDS, 198944 BYTES FREE.
39 UNUSED NODES.

==SDIR==DIRECTORY REWRITTEN  700312  003025680

==LCID==CIRCUIT NAME: CIRCUIT1, OPTIONS: 0100 FILE ONLY.
FAULTS OPTION:

==== CIRCUIT1 CIRCUIT RECORDS ====

CIRCUIT1 NODE NO.  79
$CIDS NODE NO.  71  86 BYTES, STARTING REGION NO.  12 WORD NO.  525
$LDSC NODE NO.  78  168 BYTES, STARTING REGION NO.  12 WORD NO.  547
$FMCS NODE NO.  84  132 BYTES, STARTING REGION NO.  12 WORD NO.  450
$SYMTC NODE NO.  81  124 BYTES, STARTING REGION NO.  12 WORD NO.  492
$REFS NODE NO.  82  56 BYTES, STARTING REGION NO.    12 WORD NO.  589
$SUCS NODE NO.  83   75 BYTES, STARTING REGION NO.   0 WORD NO.  20
$TERM NOT FOUND.
==== STRUCTURES LOCATIONS ====
STRUCTURE: $CIDS  OF NODE NO. 71, FETCHED
STRUCTURE: $LDSC  OF NODE NO. 78, FETCHED
STRUCTURE: $SYMTA OF NODE NO. 81, FETCHED
STRUCTURE: $REFS  OF NODE NO. 82, FETCHED
STRUCTURE: $SUCS  OF NODE NO. 83, FETCHED

86 BYTES, STARTING REGION NO. 12 WORD NO. 525
168 BYTES, STARTING REGION NO. 12 WORD NO. 547
124 BYTES, STARTING REGION NO. 12 WORD NO. 492
56 BYTES, STARTING REGION NO. 12 WORD NO. 589
75 BYTES, STARTING REGION NO. 0 WORD NO. 20

EV='01101110'B  R= 110;

6 LEADS.
2 OUTPUTS : O1 , A2
2 INPUTS : X1 , X2
0 FEEDBACKS:
0 SINGLE FAULTS.
12 SUCCESSORS IN SS.
0 TERMINALS.
6 SYMBOLIC NAMES.
0 FAUST RUNS.
3 LOGIC LEVELS.
<table>
<thead>
<tr>
<th>NO.</th>
<th>NAME</th>
<th>ATOM</th>
<th>LVL</th>
<th>INPUT</th>
<th>REFERENCES</th>
<th>FANOUT</th>
<th>OUTPUT REFERENCES</th>
<th>ASSOCIATED FAULTS NO.</th>
<th>INPUT#</th>
<th>OUTPUT#</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>X1</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>3.</td>
<td>3</td>
<td>4</td>
</tr>
<tr>
<td>2</td>
<td>X2</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>2.</td>
<td>3</td>
<td>4</td>
</tr>
<tr>
<td>3</td>
<td>A1</td>
<td>13</td>
<td>1</td>
<td>1</td>
<td>2</td>
<td>0</td>
<td>0</td>
<td>2.</td>
<td>5</td>
<td>6</td>
</tr>
<tr>
<td>4</td>
<td>N1</td>
<td>8</td>
<td>1</td>
<td>1</td>
<td>2</td>
<td>0</td>
<td>0</td>
<td>1.</td>
<td>6</td>
<td></td>
</tr>
<tr>
<td>5</td>
<td>A2</td>
<td>13</td>
<td>2</td>
<td>1</td>
<td>3</td>
<td>0</td>
<td>0</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>6</td>
<td>01</td>
<td>14</td>
<td>2</td>
<td>3</td>
<td>4</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
The second example is a description of a circuit named DECODER. Here DISP=NEW is used. The description of the circuit is not stored, because there is a circuit in the file with the name DECODER. This is evident if we look at the previous example. If we had put DISP=OLD or DISP=YES, then we would have overwritten the description presently in the file. To understand the implications of this see APPENDIX A.

Here is the run output.
**EXAMPLE 2**

```plaintext
RUNI: TRAIZE CIRCUIT=DECODER, DISP=NEW;

==FDIR==OLD DIRECTORY FETCHED.

==FETCH==TRAIZE.$OPTAB
STRUCTURE: $OPTAB OF NODE NO. 51, FETCHED 228 BYTES, STARTING REGION NO. 11 WORD NO. 621

===== STRUCTURE NO. 14 =====

SIZE: 228 BYTES, RECORD NAME: TRAIZE.$OPTAB

$OPTAB.OP#= 16;

===== OPERATORS TABLE LISTING=====

<table>
<thead>
<tr>
<th>NO.</th>
<th>SYMBOL</th>
<th>ADDRESS</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>AND</td>
<td>0</td>
</tr>
<tr>
<td>2</td>
<td>END</td>
<td>13</td>
</tr>
<tr>
<td>3</td>
<td>FBK</td>
<td>64</td>
</tr>
<tr>
<td>4</td>
<td>FBKS</td>
<td>2</td>
</tr>
<tr>
<td>5</td>
<td>FEEDBACK</td>
<td>36</td>
</tr>
<tr>
<td>6</td>
<td>INP</td>
<td>36</td>
</tr>
<tr>
<td>7</td>
<td>INPS</td>
<td>1</td>
</tr>
<tr>
<td>8</td>
<td>INPUTS</td>
<td>34</td>
</tr>
<tr>
<td>9</td>
<td>NAND</td>
<td>8</td>
</tr>
<tr>
<td>10</td>
<td>NOR</td>
<td>12</td>
</tr>
<tr>
<td>11</td>
<td>NOT</td>
<td>7</td>
</tr>
<tr>
<td>12</td>
<td>OR</td>
<td>14</td>
</tr>
<tr>
<td>13</td>
<td>DUP</td>
<td>33</td>
</tr>
<tr>
<td>14</td>
<td>DUPS</td>
<td>35</td>
</tr>
<tr>
<td>15</td>
<td>OUTPUTS</td>
<td>35</td>
</tr>
</tbody>
</table>
```
INPUTS(7) = A, B, C, D, E, F, G;
OUTPUTS(4) = ND1, ND2, ND3, ND4;
A = INP /NOR1;

***TRAIZE*** COMBINATIONAL CIRCUIT:
B = INP /NOR1, NOR3, NOR6, NOR9;
C = INP /NOR2, NOR3, NOR4;
D = INP /NOR2, NOR5, NOR8;
E = INP /NOR4, NOR7, NOR10;
F = INP /NOR5, NOR6, NOR7;
G = INP /NOR8, NOR9, NOR10;
NOR1: NOR A, B/NOR15, NOR19;
NOR2: NOR C, D/NOR12;
NOR3: NOR B, C/NOR14;
NOR4: NOR C, E/NOR13, NOR13, NOR14;
NOR5: NOR D, F/NOR17;
NOR6: NOR B, F/NOR12, NOR17;
NOR7: NOR E, F/NOR16, NOR16, NOR18;
NOR8: NOR D, G/NOR11, NOR15, NOR18;
NOR9: NOR B, G/NOR19;
NOR10: NOR E, G/NOR11, NOR20, NOR20;
NOR11: NOR NOR10, NOR8/ND1;
NOR12: NOR NOR2, NOR6/ND1;
NOR13: NOR NOR4, NOR4/ND1;
NOR14: NOR NOR4, NOR3/ND2;
NOR15: NOR NOR1, NOR8/ND2;
NOR16: NOR NOR7, NOR7/ND2;
NOR17: NOR NOR5, NOR6/ND3;
NOR18: NOR NOR8, NOR7/ND3;
NOR19: NOR NOR9, NOR1/ND4;
NOR20: NOR NOR10, NOR10/ND4;
ND1 = NAND NOR11, NOR12, NOR13;
ND2 = NAND NOR14, NOR15, NOR16;
ND3 = NAND NOR17, NOR18;
ND4 = NAND NOR19, NOR20;

END DECODER;
PHASE BUILDER
THE GIVEN CIRCUIT PARAMETERS ARE

N# = 31
E# = 31
F# = 0
CNAME = 'DECODER'

K# = 78
JOBPARM.RUNAME = 'RUN1'
JOBPARM.DISP = 'NEW';

=FDIR= OLD DIRECTORY FETCHED.

=FETCH= TRAIZE.$OPTAB
STRUCTURE: $OPTAB OF NODE NO. 51, FETCHED 228 BYTES, STARTING REGION NO. 11 WORD NO. 621

( a second listing of TRAIZE.$OPTAB has been omitted )

DECODER.$CIDS.$SUCS

CREATED.

=LEOR = THE CIRCUIT ORGANIZED AND LEVELLED IN: 4 LEVELS.

THE MAPPING:
1--  1
2--  2
3--  3
4--  4
5--  5
6--  6
7--  7
8--  8
9--  9
10-- 10
11-- 11
12-- 12
13-- 13
14-- 14
15-- 15
16-- 16
17-- 17
18-- 18
19-- 19
20-- 20
21-- 21
22-- 22
23-- 23
24-- 24
25-- 25
26-- 26
27-- 27
28-- 28
29-- 29
30-- 30
31-- 31

**STORE** FAILED, "NEW" DECODER.$CIDS EXISTS.

***TRAIZE*** FAILED TO STORE TABLES IN FILE SYSTEM.

***TRAIZE*** ERRORS FOUND BY BUILDER OR DISPOSITION = 'NO'.
NO TABLES WRITTEN INTO FILE SYSTEM.
Section 6

Error Diagnostics

Here are 2 examples of incorrect descriptions.

Example (1)

<table>
<thead>
<tr>
<th>CIRCUIT1</th>
<th>Description And Error Messages</th>
</tr>
</thead>
<tbody>
<tr>
<td>% RUN:</td>
<td>TRAIZE CIRCUIT = CIRCUIT1, DISP = NO;</td>
</tr>
<tr>
<td>INPUTS</td>
<td>(3) X1,X2,X3;</td>
</tr>
<tr>
<td>OUTPUTS</td>
<td>(2) 01,A2;</td>
</tr>
</tbody>
</table>

X1: INP;

*** TRAIZE *** COMBINATIONAL CIRCUIT.

X2: INP;

A1: AND X1,X2;

N1: NAN X1,X2;

*** TRAIZE *** ILLEGAL FUNCTION CODE ... NAN INSTATEMENT NR. 4

A2: AND X1,A1,03;

01 OR A1,N1;

*** TRAIZE *** NO LABEL ON THIS ATOM, STATEMENT NO. 6

*** TRAIZE *** ILLEGAL FUNCTION CODE ... 01 IN STATEMENT NR. 6

CONDITION ENDF OCCURRED IN STATEMENT 00010 AT OFFSET +00130 FROM ENTRY POINT NECAR.
TRAIZE has not gone through the second stage of processing. The reason is the absence of the END statement.

Omitting the END statement not only is an error, but it prevents other errors, those picked up in the second stage, from being found.

In CIRCUIT1 two errors that would be picked up in the second stage are checked.

Therefore we can not debug CIRCUIT1 in a single run.

Example (2)

In this circuit description all of the errors are found in the second stage. That is, none of the errors are syntax errors.

Therefore the error messages appear following the circuit description listing.

```
CIRCUIT2 Description

;RUN:             TRAIZE CIRCUIT=CIRCUIT2, DISP=NO;

INPUTS (3) X1,X2,X3;

OUTPUTS (1) O1;

FEEDBACKS (1) F1;

X1:             INP     /A1;

X2:             INP     /A1,O1;

A1:             AND     X1,X2 /02;

O1: OUT:        OR      A1,X2;

END CIRCUIT2;
```

Try to pick out the errors yourself before looking at the copy of error messages from TRAIZE on following page.
*** TRAIZE *** THE SUCCESSOR 02 AT POSITION 6 OF THE SUCCESSORS LIST HAS NOT BEEN DECLARED AS A LEAD OF THE CIRCUIT.

CIRCUIT2.$CIDS.$SUCS CREATED.

***TRAIZE*** THE USER-SUPPLIED SUCCESSOR FOR LEAD A1 HAS NOT BEEN DECLARED AS A LEAD.

***TRAIZE*** INPUT NAME X3 HAS BEEN GIVEN IN THE INPUT LIST BUT HAS NOT BEEN DECLARED AS AN INPUT LEAD.

***TRAIZE*** FEEDBACK NAME F1 HAS BEEN GIVEN IN THE FEEDBACKS LIST BUT HAS NOT BEEN DECLARED AS A FBK LEAD.

==LEOR==THE CIRCUIT ORGANIZED AND LEVELED IN: 3 LEVELS.

THE MAPPING:

1- > 1  2- > 2  3- > 3  4- > 4

***TRAIZE*** ERRORS FOUND BY BUILDER OR DISPOSITION = "NO".

NO TABLES WRITTEN INTO FILE SYSTEM.
GLOSSARY OF ERROR MESSAGES

Messages concerning the control card:

*** CCINT FAILED ** is printed out whenever there is any error in the
control card or control card is missing.

** CCINT ** INVALID OPERAND FOR______________________
the operand appearing after CIRCUIT or DISP
is not there.

* CONTROL CARD FOUND IN DATA
card beginning with "%" in column one was found.

. FLUSHED printed after each card appearing before the
control card. These cards are ignored.

* INVALID DELIMITER______________________
an equal sign does not appear after CIRCUIT or
DISP.

* INVALID KEYWORD______________________
either CIRCUIT or DISP do not appear in control
card and some other keyword appears in their place.

* INVALID KEYWORD IN CONTROL CARD

TRAIZE does not appear after RUNAME.
* INVALID OR MISSING OPERATION 

TRAIZE does not appear or is followed by some delimiter other than a blank.

* RUNAME INVALID OR MISSING

first name after "%" is not followed by a colon or is blank.

Messages concerning TRAIZE in general. All of these messages are preceded by *** TRAIZE ***, which will be left out here.

* ATTEMPT TO READ TRAIZE.$OPTAB FROM DIAL FILE FAILED RUN ABORTED.

This is not your error but a failure of the system.

* COMBINATIONAL CIRCUIT

This is not an error, but indicates the type of circuit you have described.

* CONTROL CARD INVALID

There was an error encountered in control card.

* COULDN'T READ CONTENTS OF COPTA FROM LIBRARY

This is not your error but a failure of the system.

* COULDN'T RECOVER $SYMTA FROM REC_DIR
* CRESSA FAILED, RUN ABORTED

The input reference leads were not written in file.
Error in input reference.

* ERRORS FOUND BY BUILDER OR DISPOSITION = "NO". NO TABLES WRITTEN INTO FILE
Could also be errors found by syntax analyzer.

* FAILED TO STORE TABLES IN FILE SYSTEM

Description of circuit not printed correctly. Check syntax errors.

* FEEDBACK NAME ___________ HAS BEEN DECLARED IN THE LIST OF FEEDBACKS BUT
ITS FUNCTION IS NOT FBK.

* FEEDBACK NAME ___________ HAS BEEN GIVEN IN THE FEEDBACKS LIST BUT HAS
NOT BEEN DECLARED AS A FBK LEAD.

* FEEDBACK STATEMENT HAS NO TERMINAL SEMICOLON, OR NR OF FEEDBKS GIVEN, = _____
IS LESS THAN NR COUNTED.

* FUNCTION NOT YET IMPLEMENTED

The function name is in the table but system is not equipped to handle it.

* ILLEGAL FUNCTION CODE ___________ IN STATEMENT NR ___________.

* INPUT NAME ___________ HAS BEEN GIVEN IN THE INPUT LIST BUT HAS NOT BEEN
DECLARED AS AN INPUT LEAD.
* INPUT NAME ________ HAS BEEN GIVEN IN THE LIST OF INPUTS BUT ITS FUNCTION IS NOT INF.

* INPUT REFERENCE LIST FOR STATEMENT NR_______ DOES NOT END WITH SLASH OR SEMICOLON.

* INPUTS, OUTPUTS AND FEEDBACKS STATEMENTS MUST PRECEDE ALL LEAD DECLARATIONS.

* INPUTS, OUTPUTS AND FEEDBACKS STATEMENTS CANNOT APPEAR IN THE MIDDLE OF LEAD DECLARATIONS.

* INPUTS STATEMENT HAS NO TERMINAL SEMICOLON, OR NR OF INPUTS GIVEN, =______

  IS LESS THAN NR COUNTED =__________.

* INVALID KEYWORD IN INPUTS, OUTPUTS OR FEEDBACKS STATEMENT__________.

  The keyword was not found in table-check spelling.

* NO INPUT REFERENCES WERE FOUND FOR STATEMENT NR__________.

  Input references for each lead are necessary.

* NO INPUTS STATEMENT FOUND.

  Inputs statement missing or in wrong place.

* NO LABEL ON THIS LEAD STATEMENT NR.=__________

* NO OUTPUTS STATEMENT FOUND.

  Outputs statement missing or in wrong place.

* NO PARENTHESIS FOUND AROUND NUMBER OF FEEDBACKS IN FEEDBACKS STATEMENT.

  Following "FEEDBACKS" must have number of feedbacks in parenthesis.
* NO PARENTHESES FOUND AROUND NUMBER OF INPUTS IN INPUTS STATEMENT.
   Following "INPUTS" must have number of inputs in parenthesis.

* NO PARENTHESES FOUND AROUND NUMBER OF OUTPUTS IN OUTPUTS STATEMENT.
   Following "OUTPUTS" must have number of outputs in parenthesis.

* NO TERMINAL SEMICOLON.

* NUMBER OF FEEDBACKS GIVEN = _________ IS MORE THAN NUMBER ACTUALLY
   COUNTED _________.

* NUMBER OF INPUTS GIVEN = _________ IS MORE THAN NUMBER ACTUALLY
   COUNTED _________.

* NUMBER OF OUTPUTS GIVEN = _________ IS MORE THAN NUMBER ACTUALLY
   COUNTED _________.

* OUTPUT NAME _________ HAS BEEN GIVEN IN THE OUTPUT LIST BUT HAS NOT
   BEEN DECLARED AS AN OUTPUT LEAD.

* OUTPUTS STATEMENT HAS NO TERMINAL SEMICOLON OR NR OF OUTPUTS GIVEN =
   _________ IS LESS THAN NR COUNTED.

* THE FEEDBACK LEAD OF STATEMENT _________ HAS MORE THAN 1 INPUT REFERENCE.
   A feedback lead can have and must have one input reference.

* THE INPUT _________ OF THE LEAD _________ HAS NOT BEEN DECLARED AS A
   LEAD OF THE CIRCUIT.
* THE SINGLE INPUT REFERENCE FOR THE FEEDBACK LEAD OF STATEMENT NR________

            IS FOLLOWED BY / OR ;

* THE SUCCESSOR_________AT POSITION_________OF THE SUCCESSORS LIST HAS

            NOT BEEN DECLARED AS A LEAD OF THE CIRCUIT.

            Check out put reference names for spelling and input
            reference names.

* THE USER SUPPLIED SUCCESSOR_________FOR LEAD_______IS IN DISAGREEMENT

            WITH THE COMPUTED ONES. THEY ARE ... ________________

* THIS STATEMENT HAS A SLASH BUT NO OUTPUT REFERENCES CAN BE FOUND.

* TOO MANY INPUT REFERENCES FOR THE LEAD OF STATEMENT NR. ____________
APPENDIX A

Other values for DISP

The following are alternative values for DISP. The user should become familiar with the system's file operation before he commands anything other than DISP=NO.

DISP=NO - do not save the run results in the file system.
DISP=YES - save the run results in the file system.
DISP=NEW - create new entries in the file for the run results.
DISP=OLD - overwrite already existing records of run results with the same name.

Notice that YES is a combination of OLD and NEW, that is, if there are records overwrite, or if there are no records create new entries.