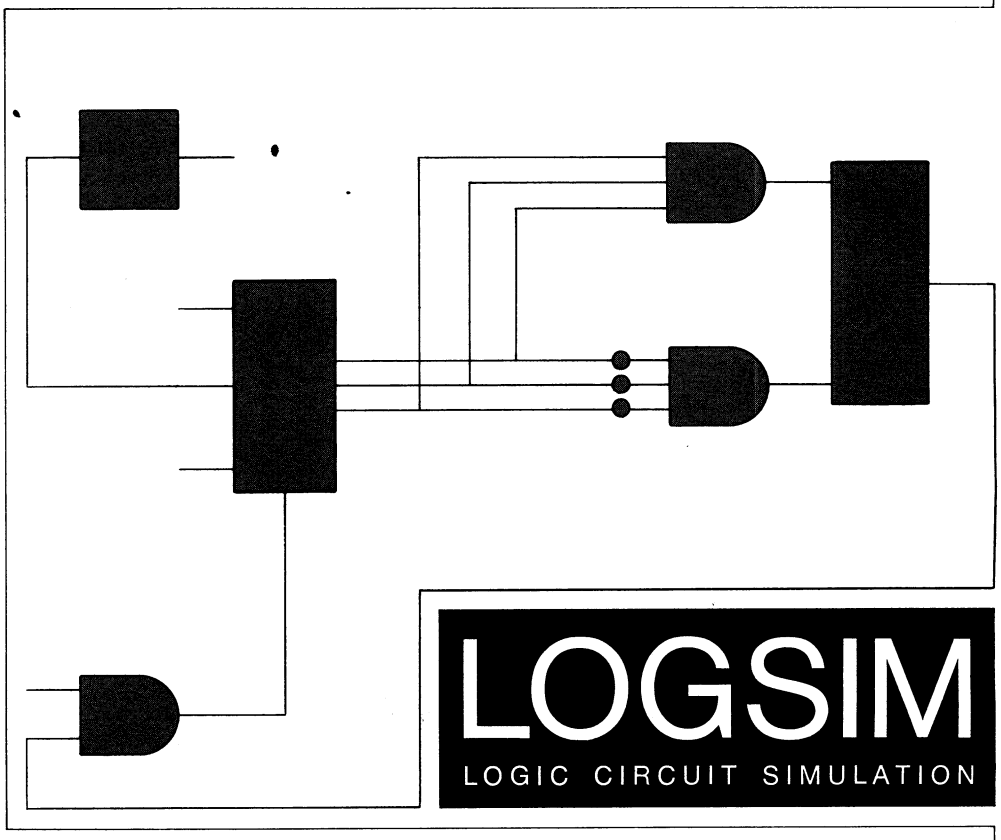


LOGSIM

LOGIC CIRCUIT SIMULATION

TYMSHARE®

Price: $2.00

TYMSHARE MANUALS

REFERENCE SERIES

LOGSIM

Logic Simulation Program

February 1970

Proprietary Program Series

Tymshare, Inc.
525 University Avenue, Suite 220
Palo Alto, California

TABLE OF CONTENTS

Section Page

I. INTRODUCTION

Logic Simulation With LOGSIM

The Tymshare Application Program, LOGSIM, allows a user to simulate a digital logic circuit on a time-shared computer. LOGSIM allows the user to check for logical errors and timing problems without first building a "breadboard" model. The logic circuit is described using built-in logic blocks with user specified propagation delay and interconnections. The engineer can use this program to generate timing diagrams, trace every logical change of each block, make small or large changes in the circuit configuration, and introduce stray propagation delays before actually breadboarding the circuit.

LOGSIM Capabilities

Use of up to six character mnemonic block names.

Use of up to 250 blocks per simulation.

Timing diagrams over the time ranges specified by the user. Allowable time range is 0 to 8388607.

A complete TRACE of logical changes within the circuit over a specified time range.

29 built in types of logic blocks.

The User May:

Describe the circuit configuration via a file or the terminal.

Modify the circuit under user control by:

Altering or deleting existing blocks
Adding or inserting new blocks

List part of or all of the present circuit configuration.

Specify the assumed logical state of any block.

Examine the orginal and present octal word of a serial word generator.

Determine the current logical status of part or all of the present logical blocks

Save the present circuit configuration on a file for later use

Save the present logical status of the circuit for continuation at another date

Cycle up to 5 input switch registers over all possible combinations

Built-in Logical Blocks

Single Block Descriptions

Inverters
AND gates
OR gates
Exclusive OR gates
NAND gates
NOR gates
RS flip flops
JK flip flops
RST flip flops
Single shot multivibrators
Clocks
Threshold logic gates
Time generators
Serial word generators
Counter blocks
Shift register blocks
Full adders

Variable Length Blocks

Gate registers
OR registers
UP/DOWN counters
SHIFT registers
PRESET UP/DOWN counters

Special Blocks

Output modules
Trigger modules
Stop blocks
Null blocks

II. LOGSIM TERMINOLOGY AND CONVENTIONS

The following standard conventions will be used in this manual.

Coupling Terminology

Edge triggered and direct coupled blocks will be indicated as in figure 1.

Direct coupled input Edge triggered input

Figure 1. Coupling Terminology

Each block is referenced by a mnemonic name. The program may contain up to 250 blocks.

Mnemonic Name Rules:

1. Each name may be from one to six characters in length. All characters over six are truncated.

2. The names may contain any combination of alphabetic characters A-Z and digits 0-9. All combinations are possible except those discussed below.

3. RESERVED NAMES

   Following are reserved names and symbols that cannot be used as block names.

T	true
F	false
*	reserved for switch identification
ALL	command designation
-	minus sign reserved for signifying block complements
+	special block name
$	last block number

4. Special Case Names

Those names referred to as inputs to GAT or ORR register must have the first three characters of the name unique, since the generated block inputs are referred to as XXX+1, XXX+2, etc. (see example 6)

Block Conventions

The output of each block with the exception of special blocks will be either a logical true or false. Multiple output devices are constructed by the program from individual blocks.

The input to a block may be complemented by referencing the input block name as the negative of the input block name.

Direct coupled blocks will change logical state if the correct inputs are applied and only if these inputs remain for at least the propagation delay of the block. The change of state in the output will occur after the propagation delay time has elapsed. If the inputs do not remain constant for the full propagation delay period, the state will not change.

Triggered blocks will assume the correct logical state, one propagation delay interval after the input conditions are satisfied. This delay begins with a change of state of the trigger.

Edge triggered blocks normally trigger on the rising edge of the input. Falling trigger blocks are obtained with the complement of the input trigger block number.

The terms true and false, 1 and 0, and high or low are the same respectively and are used interchangeably.

The following Boolean logic symbols will be used:

Asterisk (*) for "and", plus (+) for "or", and an overline (¯) for the complement.

In the examples the following conventions will be used:

"_____" will denote items typed by the user

 will denote alt mode or ESCAPE

 will denote carriage return

 will denote line feed

III. SUMMARY OF COMMANDS

The following information can be printed to the terminal by typing the underlined command. A more detailed explanation of each command is given in Section IV.

:HELP

COMMAND	DESCRIPTION
HELP (OR) ?	REPRINTS THIS LIST.
CAPABILITIES	DESCRIBES PROGRAM CAPABILITIES.
INSTRUCTIONS	HOW TO EXECUTE THE PROGRAM.
CHARGES	
CREDITS	IMPLEMENTED BY TYMSHARE
VERSION	LATEST UPDATE
SAMPLE	SAMPLE PROBLEM RUN WITH LOGSIM.
INPUT	DELETES CIRCUIT. ENTER A NEW CIRCUIT.
ADD	ADD BLOCK(S) TO CIRCUIT.
DELETE	DELETES ANY OR ALL CIRCUIT BLOCKS.
INSERT	INSERT A BLOCK IN FRONT OF ANY BLOCK.
ALTER	REPLACE A BLOCK WITH NEW BLOCK(S).
INITIALIZE	REINITIALIZES ALL BLOCKS AND SWITCH REGISTERS TO TO THE UNDEFINED STATE.
PRINT	PRINTS TIMES FOR CHANGE OF STATE.
LIST	LISTS ANY OR ALL BLOCKS.
SAVE	SAVES THE CIRCUIT ON A FILE.
DUMP	DUMPS THE PRESENT STATUS OF A CIRCUIT SIMULATION FOR RECOVERY LATER.
RECOVER	ALLOWS RESUMPTION OF SIMULATION FROM DUMP FILE.
SET	SETS SPECIFIED BLOCKS TO LOGICAL TRUE.
RESET	RESETS SPECIFIED BLOCKS TO LOGICAL FALSE.
WRITE	WRITE A NEW OCTAL WORD INTO THE SERIAL WORD GENERATOR.
READ	PRINTS THE OCTAL WORD OF THE SERIAL WORD GENERATOR.

CYCLE	SIMULATES A CIRCUIT FOR A SPECIFIED TIME PERIOD WITH SWITCH REGISTERS (5 MAX.) CYCLED THRU EVERY POSSIBLE COMBINATION OF TRUE AND FALSE.
TRACE	STARTS SUMULATION SUCH THAT FOR EVERY LOGICAL CHANGE, DURING SIMULATION, THE BLOCK NAME, THE NEW LOGICAL STATE AND THE TIME OF THE STATE CHANGE WILL BE LISTED TO THE TERMINAL.
RUN	SIMULATES CIRCUIT AND PRINTS ANSWERS ON TERMINAL.
QUIT (OR) Q	QUITS TO EXEC.

ANY OF THESE COMMANDS MAY BE SHORTENED TO THE FIRST THREE LETTERS EXCEPT 'INSTRUCTIONS' AND 'INSERT' WHICH REQUIRE AT LEAST THE FIRST FOUR LETTERS.

:CAPABILITES

THIS PROGRAM ALLOWS THE LOGIC DESIGN ENGINEER TO SIMULATE A LOGIC CIRCUIT BEFORE BUILDING A BREADBOARD. THIS SIMULATION INCLUDES THE CAPABILITY TO SPECIFY PROPAGATION DELAYS THROUGH THE LOGIC ELEMENTS CHOSEN. THE OUTPUT OF THE PROGRAM CAN BE:

1. A TIMING DIAGRAM OF UP TO 15 DIFFERENT LOGICAL BLOCKS WHICH WILL BE GENERATED AT TIMES AS SPECIFIED BY USER.

2. A TRACE OF EVERY LOGICAL CHANGE IN THE CIRCUIT OVER SOME SPECIFIED TIME RANGE.

THE PROGRAM CAN HANDLE UP TO 250 LOGIC BLOCKS. AN EXTENSIVE COMMAND SET ALLOWS THE USER TO INPUT CIRCUIT DESCRIPTIONS FROM THE TERMINAL OR A FILE.

THE CIRCUIT CONFIGURATION CAN BE MODIFIED ON-LINE BY THE USER. A MODIFIED CIRCUIT MAY BE STORED ON A FILE.

AFTER COMPLETION OF A TIME PERIOD, A DUMP FILE MAY BE SAVED TO RESUME SIMULATION AT SOME LATER DATE.

PROGRAM ERROR DIAGNOSTICS HELP THE USER WITH ON-LINE DEBUGGING.

:INSTRUCTIONS

THE FOLLOWING ARE BUILT IN LOGICAL BLOCKS WHICH THE USER MAY SELECT FOR THE SIMULATION. OTHER BLOCKS CAN OF COURSE BE MADE OF THE BUILT-IN BLOCKS BY THE USER. THE BLOCKS ARE DESCRIBED IN THE FORM SHOWN BELOW.

#,TYPE,PD,I1,I2,...IN

- # - BLOCK NAME (1 TO 6 CHARACTERS IN LENGTH)
- TYPE - TWO OR THREE LETTER DESCRIPTOR OF THE BLOCK TYPE
- PD - PROPAGATION DELAY WHERE PD MUST BE AN INTEGER AND 0<=PD<=8388607
- I1 - IN - INPUT PARAMETER LIST WHERE THE LIST MAY BE IN GENERAL
  1. BLOCK NAMES
  2. TRUE OR FALSE (T OR F)
  3. SWITCH REGISTERS (*1 THRU *36)

INVERTER	#,INV,PD,I
AND GATE	#,AND,PD,I1,...,I5
OR GATE	#,OR,PD,I1,...,I5
EXCLUSIVE OR GATE	#,EOR,PD,I1,I2
NAND GATE	#,NAN,PD,I1,...,I5
NOR GATE	#,NOR,PD,I1,...,I5
R S FLIP FLOP	#,RS,PD,SET,RESET
J K FLIP FLOP	#,JK,PD,TRIGGER,J,K,SET,RESET
RST FLIP FLOP	#,RST,PD,TRIGGER,SET,RESET
SINGLE SHOT MULTIVIBRATOR	#,SS,PD,PULSE WIDTH,TRIGGER
CLOCK	#,CLK,PERIOD/2,ON/OFF
TIME GENERATOR	#,TIM,TIME
SERIAL WORD GENERATOR	#,SWG,PD,TRIGGER,RESET,OCTAL WORD
FULL ADDER	#,ADD,PD,I1,I2,CARRY IN
THRESHOLD GATE	#,THR,PD,THL #,I1,...,I4
GATE REGISTER	#,GAT,PD,N,I,GATE
OR REGISTER	#,ORR,PD,N,I1,...,I4
UP/DOWN COUNTER	#,CNT,PD,N,T,S,R,UP/DOWN
PRESET UP/DOWN COUNTER	#,PRE,PD,N,T,SET DATA,SET CTRL,UP/DOWN
SHIFT REGISTER	#,SFT,PD,N,T,RGHT DATA,LFT DATA,R,R/L
OUTPUT MODULE	#,OUT,PD,T,I1,...,I15
TRIGGER MODULE	#,TRG,PD,I1,...,I15
STOP BLOCK	#,STO,PD,I
NULL BLOCK	#,NUL,0

```
:SAMPLE

-LOGSIM

:INPUT T


BLK  DES: A,INV,1,*1
BLK  DES: B,INV,2,*2
BLK  DES: C,AND,3,A,B
BLK  DES: D,OUT,2,E,A,B,C
BLK  DES: E,TRG,0,A,B,C
BLK  DES: END

:LIST ALL

A        INV          1  *1
B        INV          2  *2
C        AND          3  A        B
D        OUT          2  E        A        B        C
E        TRG          0  A        B        C


: CYCLE *1-*2
TRACE ? N
OK
START TIME = 0
FINISH TIME = 10


* 1=F
* 2=F
   BLOCK    A    B    C
   NAME
      TIME  01   01   01
         OS 0    0    0
         3    1    1   0
         5F   1    1    1
         5   STOP AT STEADY STATE
```

```
* 1=T
* 2=F
  BLOCK    A    B    C
  NAME
     TIME  01   01   01
        0S 0    0    0
        2F 0     1   0
        2  STOP AT STEADY STATE

* 1=F
* 2=T
  BLOCK    A    B    C
  NAME
     TIME  01   01   01
        0S 0    0    0
        1F  1   0    0
        1  STOP AT STEADY STATE

* 1=T
* 2=T
  BLOCK    A    B    C
  NAME
     TIME  01   01   01
        0S 0    0    0
        0F 0    0    0
        0  STOP AT STEADY STATE

:QUIT

-
```

IV. OPERATING INSTRUCTIONS

LOGSIM allows the user to simulate a logical circuit using the following logical elements.

250 Logical Blocks

Each block is referenced by a block name. The block state may be set to a state consistent with the device description. All blocks are initially set to a logical false condition. The initial state of the block can be changed under user control. The run phase of the program will set all of the block states to be consistent with the circuit logic.

Switch Registers

There are 36 one bit registers which can be set to any combination of logical true or false. They are referenced by an asterisk (*), followed by a register number. These registers may be referenced as the complement with a negative register number. For example, -*5 is the complement of switch register *5. These registers may be initially set to the true state by the user.

Switch Register Cycling

The user can elect to cycle from one to five switch registers through all possible combinations of states.

Output Display Module

The results of the simulation can be printed on the terminal in a manner similar to a 15 channel sampled chart recorder. An OUT block may be specified in the circuit configuration. If no OUT block is specified, LOGSIM will automatically pause during the run mode to allow the user to select the desired output. If more than one OUT block is described in the circuit configuration, only the first one in the list is used

Logical Constants

The logical constants of T or F for true and false respectively are used for fixed input logical blocks.

LOGSIM gives the user the following system capabilities:

File INPUT and SAVE

The user can INPUT new logical circuits, SAVE the present circuit configuration, DUMP the current circuit status, and RECOVER to run at a later date.

Terminal Input

A new logical circuit may be INPUT by the user or modifications made on the existing circuit from the terminal. The user can SET and RESET logical status, use the output display module to observe state changes, and perform user control of the program operation.

LOGSIM Data Preparation

The digital logic circuit must be composed of the built-in standard blocks. Non-standard blocks may be simulated by combining existing standard blocks.

Each block is assigned a mnemonic name. The selection of block names is restricted by the duplication of user assigned and program assigned names. The multiple range should not duplicate a user assigned block name. An error diagnostic will print out if this case occurs.

The data may be entered from a file or input directly into the program from the terminal. Examples of both types of inputs are shown in Section VII.

User Commands in LOGSIM

All user input data is underlined in this users manual. All user responses are followed by a carriage return.

LOGSIM is called by simply typing in the Tymshare EXECUTIVE:

-LOGSIM

The program responds with the colon indicating the user is in the LOGSIM command mode.

COMMAND MODE

The command mode allows the user to specify a variety of options to perform the program operations. After each operation, the program returns to the command mode to wait for further instructions.

The following command options are available in LOGSIM. (A short table of these commands is listed in Section III for easy reference). All commands can be abbreviated to the first three letters except INSTRUCTIONS and INSERT which require four.

:HELP -this command prints a list of all possible commands (shown below) with a brief description.

HELP or ?	CAPABILITIES	INSTRUCTIONS	CHARGES	CREDITS
VERSION	INPUT	ADD	DELETE	INSERT
ALTER	PRINT	LIST	SAVE	DUMP
RECOVER	SET	RESET	WRITE	READ
CYCLE	TRACE	RUN	QUIT or Q	
SAMPLE	INITIALIZE			

Circuit Input and Save Commands

The following group of commands is used to enter a circuit from the terminal or a file.

When the input is from the terminal, a request is made for BLK DES: which requires a block description in the form discussed in Section V.

Error messages will print out directly following the block description when the description is incorrect. During input from a file, the descriptions as well as error messages will be printed.

Blocks with errors will not be retained in the circuit configuration. These blocks may be re-entered after the error message when input is from the terminal, or entered from the terminal after the file reading is complete.

:INPUT file name or T (terminal)

or

:INPUT↵

INPUT FROM: file name or T

BLK DES: #, Type, PD, $I_1,...I_n$ - allows the user to enter block descriptions from the terminal. The INPUT command deletes any previous circuit configuration in the program. Section VII contains a list of possible error messages. To return to the command mode type END for the BLK DES:.

:ADD file name or T

or

:ADD↵

INPUT FROM: file name or T

BLK DES: #, Type, PD, $I_1,...I_n$ - allows the user to add blocks to the existing circuit. To return to the command mode type END for the BLK DES:.

:**SAVE** file name or T

or

:**SAVE**,

OUTPUT TO: file name or T
OLD FILE or **NEW FILE**

- allows the user save the present circuit configuration on a file for later input using the INPUT command.
After OLD FILE or NEW FILE prints,
1. To verify writing to an old or new file type a carriage return.
2. To avoid writing the file, type an ALTMODE or ESCAPE

:**DUMP** file name or T

or

:**DUMP**,

OUTPUT TO: file name or T
OLD FILE or **NEW FILE**

- allows user to capture the present logic status and configuration of the circuit for recovery and continuation of analysis at a later date.
After OLD FILE or NEW FILE prints,
1. To verify writing to an old or new file type a carriage return.
2. To avoid writing the file, type an ALTMODE or ESCAPE.

The DUMP command should be used to save any circuit which may be simulated or changed and simulated at a later date with LOGSIM.

The DUMP file is a binary file and will require only 3 to 4 CPU seconds to load with the RECOVER command; whereas, even small circuits in symbolic form will generally require more CPU time when loaded with the INPUT command. The DUMP file will always contain 13824 characters.

:**RECOVER** file name or T

or

:**RECOVER** ↵

INPUT FROM: file name or T

- allows the user to recover a file created with the DUMP command.

Circuit Examination and Alteration Commands

The following commands allow the user to examine or make changes in the circuit. The commands require reference to block names and can be used in several forms.

1. Program Prompting

   When the command name is followed by a carriage return, a request will be made for BLK NAMES: or BLK NAME:. The name(s) may be:

:LIST,

BLK NAMES: A (single block)

or

BLK NAMES: A,B,G (single blocks)

or

BLK NAMES: A,B-D,G (single and ranges)

or

BLK NAMES:, (all blocks)

or

BLK NAMES: ALL (all blocks)

or

BLK NAMES: $ (last block)

2. Directly following Command (for advanced users)

Any of the forms shown below are allowed. The request for BLK NAMES: is not printed when the list of block names is typed directly after the command.

:LIST A	(single block)
or	
:LIST A,B,G	(single blocks)
or	
:LIST A,B-D,G	(single and ranges)
or	
:LIST ALL	(all blocks)
or	
:LIST $	(last block)

The commands followed by

B1, B2,...Bn, ALL, or $

indicate that the command can be of the two forms shown above

:LIST Bl, B2,...Bn, ALL, or $
or

:LIST,

BLK NAMES: Bl, B2,...Bn, ALL, or $

- lists to the terminal the present circuit configuration. The values of B may be selected block names or ranges. Individual and name ranges may be listed consecutively.

:DELETE Bl, B2,...Bn, ALL, or $
or

:DELETE,

BLK NAMES: Bl, B2,...Bn, ALL, or $

- allows the user to delete any or all blocks. Note: The DELETE command will not delete created blocks by specifying only the original block name. (i.e. ADD or variable length blocks) Specifying the name of the ADD block will not delete the AD* block. Deletion of created blocks must be explicitly defined.

:**INSERT** single block

or

:**INSERT**,

BEFORE BLK NAME: single block

- allows the user to insert new blocks in front of any existing block in the circuit.

:**ALTER** single block

or

:**ALTER**,

BLK NAME: single block

- allows the user to replace a single block in the circuit with one or more new blocks. This command combines the use of a DELETE and an INSERT.

Note: The ALTER command will not delete created blocks by specifying only the original block name. The alteration of these block types (i.e. ADD and variagle length blocks) requires all blocks to be deleted and a replacement made with the INSERT command. ALTER may be performed on individual created blocks.

State Initialization Commands

These commands allow the user to set the initial state of blocks and switch registers to logical true or false and examine the states. The block and switch register states are undefined (U) until either the user sets the state or the program analysis begins. All undefined blocks and switch registers are initially set to logical false at the beginning of an analysis. An examination of the block state may be made with the PRINT command.

These commands can be performed on switch registers; thus B1, B2,...Bn now can include *N for B. (where * indicates a switch register and N, the switch number). Switch register numbers may be specified as single blocks with block names but can not be mixed in ranges, i.e., (*N-B) is illegal.

:**SET** B1, B2,...Bn, ALL, or $

or

:**SET**,

BLK OR SW: B1, B2,...Bn, ALL, or $

- allows the user to set blocks and switch registers to the state of logical true.

:**RESET** B1, B2,...Bn, ALL, or $

or

:**RESET**,

BLK OR SW: B1, B2,...Bn, ALL, or $

- allows the user to reset blocks and switch registers to the state of logical false.

:PRINT Bl, B2,...Bn, ALL, or $

or

:PRINT,

BLK OR SW: Bl, B2,...Bn, ALL, or $

- lists to the terminal the present logical status of blocks and switch registers. Also listed is the last time a change of state occurred and the predicted time for the next change of state for logical blocks.

:INITIALIZE

- this command sets all blocks and switch registers to an undefined state. This condition is the same as the state immediately after entering a new circuit.

Serial Word Generator Commands

The following commands allow the user to examine the original and the present octal word in the serial word generator and to write a new octal word into the serial word generator memory.

The command set for specifying block names is the same as other commands except only those blocks which are serial word generators will be printed on the terminal.

:**READ** B1, B2,...Bn, ALL or $

or

:**READ**,

BLK NAMES: B1, B2,...Bn, ALL or $

- allows the user to examine the original and the present octal word in a serial word generator.

:**WRITE** B1, B2,...Bn, ALL or $

or

:**WRITE**,

BLK NAMES: B1, B2,...Bn, ALL or $

- allows the user to write a new octal word in the serial word generator memory.

Circuit Analysis Commands

The following commands are used to control the simulation analysis of the circuit.

```
:RUN
START TIME = XX
FINISH TIME = XX
```

- allows the user to begin circuit simulation for the range of time specified. The time range can be between 0 and 8388607.

```
:TRACE
OK
START TIME = XX
FINISH TIME = XX
```

- allows the user to begin simulation in the trace mode for the range of time specified. The trace mode prints out each block and its new state every-time the block changes state.

```
:CYCLE  S1, S2,...Sn        (switch register numbers
  or                         with or without *)

:CYCLE,

SWITCH NUMBER: S1, S2,...Sn
TRACE? (Y or N),
OK
START TIME = XX
FINISH TIME = XX
```

- the user specifies switch number to be used in the CYCLE mode. A maximum of five switches can be cycled through all possible combinations during the run. This command is discussed in more detail in Section VI.

:**QUIT**
or

:**Q** - allows the user to exit from LOGSIM and return to the Tymshare EXECUTIVE mode. The present circuit configuration in LOGSIM at this time is deleted unless saved by either the SAVE or DUMP command.

Initial Conditions of Circuit

During the initial definition of the circuit either from a file or the terminal, all switches and block states are undefined. This can be seen with the use of the PRINT command. Any blocks or switch registers can be SET to logical true or RESET to logical false before any of the commands CYCLE, TRACE, or RUN are executed. At run time, all blocks and switches which were not set to true or false, will be set to logical false.

At run time a copy of the initial conditions of the circuit is saved for later reinitialization. This will occur if the new start time for a second simulation run is less than the finish time of the previous run. Reinitialization also occurs for each cycle of the cycle command.

IMPORTANT If the user has completed one or more time periods with the RUN or TRACE command and gives any command except RUN or TRACE, LOGSIM will contain two sets of circuit state conditions.

State 1. Initial circuit state conditions with all blocks either at logical true as set by the user or to false as set by LOGSIM.

State 2. The circuit state at the end of last run period.

If the user has completed one or more simulation time periods and requires a change in the initial conditions of State 1, to start at time zero, the following sequence of commands must be performed.

```
:INITIALIZE
OK

: perform commands to set new initial conditions
```

Consider the following cases:

Case 1.

```
:RUN
START TIME = 0
FINISH TIME = 10
```

Initial conditions (State 1) saved here.

```
    .
    .
    .

:SET A
OK
```

Setting block A to true.

```
:RUN
START TIME = 11
FINISH TIME = 50
```

Initial conditions (State 2) at time 10 with block A set to true.

Case 2. INCORRECT

```
:RUN
START TIME = 0   Initial condition (State 1) saved here
FINISH TIME = 10

    .
    .
    .

:SET A
OK


:RUN
START TIME = 0   Starting at 0 reinitialized with State 1
FINISH TIME = 10 initial conditions and SET A was ignored.
```

Case 3. CORRECT

```
:RUN
START TIME = 0   State 1 saved here
FINISH TIME = 10

    .
    .
    .

:INITIALIZE      This command indicates to LOGSIM
OK               that new initial conditions will
                 be needed.

:SET A
OK


:RUN
START TIME = 0   A new set of initial conditions (State 1)
FINISH TIME = 10 will be saved here.
```

V. BLOCK DESCRIPTIONS

Block descriptions of every logical block in the circuit to be simulated are required from the user to enable the computer to analyze the problem. Each block description has the general form:

#, TYPE, PD, I_j,...I_n

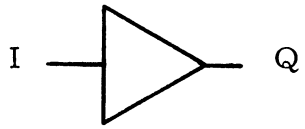
- # - Block name (see section II for block name conventions)
- TYPE - Two or three letter descriptor of the block type
- PD - Propagation delay which must be in integer units ranging in value from 0 to 8388607. It is necessary to maintain a common system of units for all delays and time units throughout one simulation.

I_j through I_n represent the input parameter list for the particular block. Any "I" could be either a block number, switch number, a logical constant (T or F), or just a number, each interpretation depending upon the type of block being described. Most blocks require a specified number of parameters to be defined; though some blocks do allow a variable number of inputs to be defined.

Single Block Descriptions

This section will show the single logical blocks, the required description of that block configuration, a logic symbol of the block, and a logical definition of the block.

INVERTER

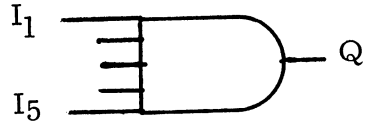


#, INV, PD, I

$Q = \overline{I}$

The inverter should be used only when inversion is required with a propagation delay. A minus sign preceding the input would give one inversion without a propagation delay.

AND

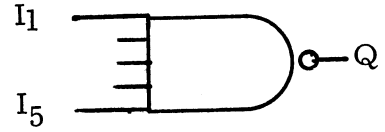


#, AND, PD, I_1, I_2, I_3, I_4, I_5

$Q = I_1^* I_2^* I_3^* I_4^* I_5$

Any combination of one thru five inputs is legal.

NAND

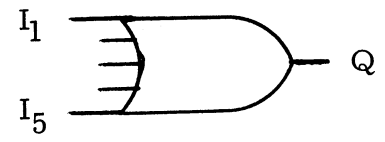


#, NAN, PD, I_1, I_2, I_3, I_4, I_5

$Q = \overline{I_1^* I_2^* I_3^* I_4^* I_5}$

Any combination of one thru five inputs is legal.

OR

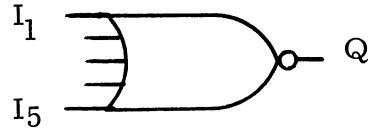


#, OR, PD, I_1, I_2, I_3, I_4, I_5

$Q = I_1 + I_2 + I_3 + I_4 + I_5$

Any combination of one thru five inputs is legal.

NOR

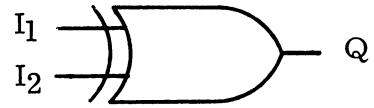


#, NOR, PD, I_1, I_2, I_3, I_4, I_5

$$Q = \overline{I_1 + I_2 + I_3 + I_4 + I_5}$$

Any combination of one thru five inputs is legal.

EXCLUSIVE OR

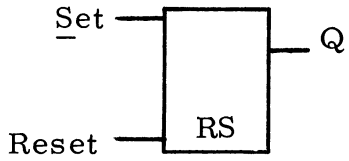


#, EOR, PD, I_1, I_2

$$Q = (I_1 * \overline{I_2}) + (\overline{I_1} * I_2)$$

RS FLIP-FLOP

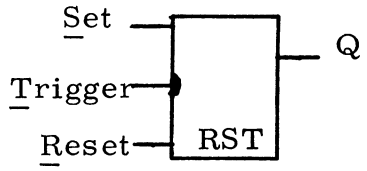
#, RS, PD, S,R



SR \ Q	0	1	
00	0	1	old state
01	0	0	new state
10	1	1	
11	U	U	

Undefined

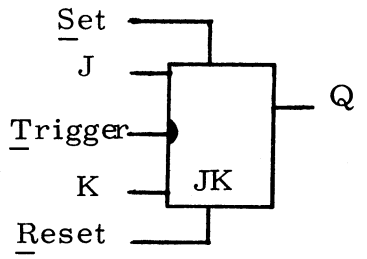
RST FLIP-FLOP



#, RST, PD, T, S, R

The R and the S inputs obey the same truth table as the RS Flip-Flop. If either R or S is high, the trigger will be inhibited. If both R and S are low and T changes from low to high, Q will reverse state.

JK FLIP-FLOP

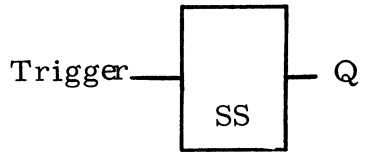


#, JK, PD, T, J, K, S, R

The R and the S input obey the same truth table as the RS Flip-Flop. If either R or S is high, then T, J, and K are inhibited. If both R and S are low and T changes from low to high, Q will obey the following truth table.

JK \ Q	0	1	old state
00	0	1	
01	0	0	new state
10	1	1	
11	1	0	

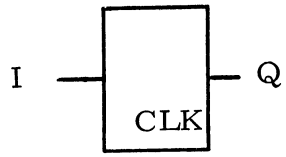
SINGLE SHOT



#, SS, PD, Pulse width, T

Q will change from low to high for the duration of the pulse width when initiated by T changing from low to high.

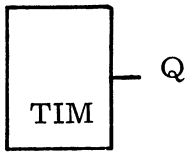
CLOCK



#, CLK, Period/2, I

Q will be a square wave of the specified period when I is high. When I is low, the clock will be turned off and Q will be high.

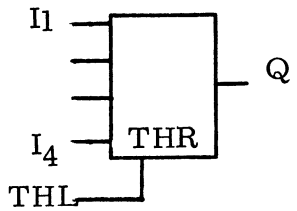
TIME



#, TIM, Time

Q will become high when time in the simulation reaches the preset time. TIM is useful to initiate action within the circuit at a specified time.

THRESHOLD GATE



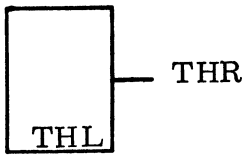
#, THR, PD, THL#, I_1, I_2, I_3, I_4

Q will become high if the summed weight level exceeds a threshold. If an input is high it has a level of 1; otherwise, 0. Each input has a weight W associated with it, which along with the threshold level is defined by the name THL#. Thus Q is high only if

$(I_1 * W_1) + (I_2 * W_2 + (I_3 * W_3) + (I_4 * W_4)$ = the threshold level where * means multiplication and + means addition. The user must specify a THL# block for each THR.

Any combination of one thru four inputs is legal.

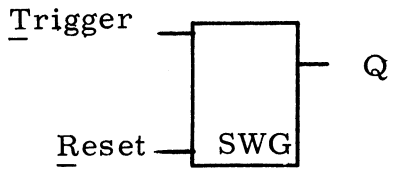
THRESHOLD LEVELS



#, THL, Threshold Level, W_1, W_2, W_3, W_4

This block is used in conjunction with a threshold gate to specify the threshold level and the input weights of that threshold gate. The number of weights specified for the THL must be the same as the number of inputs for the THR. The level may be an integer between 0 and 8388607 while the weights may be integers values between 0 and 250.

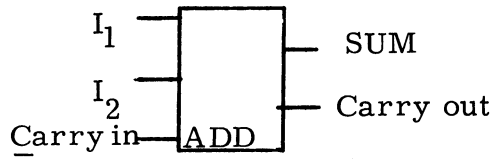
SERIAL WORD GENERATOR



#, SWG, PD, T, R, Octal Word

Each time the T input changes from low to high the lower order bit is shifted to the block output Q. After the entire word is shifted out, further triggers will cause the last bit to be repeated. When R goes from low to high, the original octal word is reset in the generator memory.

FULL ADDER



#, ADD, PD, I_1, I_2, C

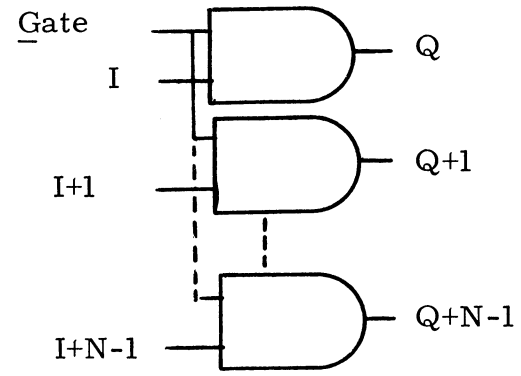
This device will produce the sum of two binary numbers I_1 and I_2 with C the carry into an AD* block according to the following truth table. The carry bit is stored in the created block AD* with the name XXX+1 where XXX is the first three characters of the ADD name.

I_1	I_2	C_{in}	Sum	$Carry_{out}$
0	0	0	0	0
0	0	1	1	0
0	1	0	1	0
0	1	1	0	1
1	0	0	1	0
1	0	1	0	1
1	1	0	0	1
1	1	1	1	1

Variable Length Blocks

The following blocks are of variable length. A single block description will produce a series of interconnected devices. If a block of length N is to be specified (where N appears in the block description) the first block is defined by the user and the remaining N-1 devices will be created by LOGSIM. Up to 33 devices may be generated by one block description.

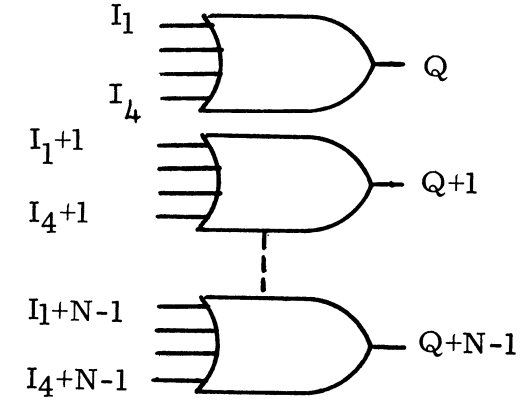
GATE REGISTER



#, GAT, PD, N, I, G

This gate is used to switch N inputs where the first input switched is specified as I. When G becomes high, the consecutive blocks, I to I+N-1 will be gated. (If G is high, Q=I, Q+1=I+1, ..., Q+N-1=I+N-1) G inhibits when low. (If G is low, Q, Q+1, ..., Q+N-1 = low) A gate register generates N "AND" gates for the simulation and would be shown as AND gates if command LIST were typed.

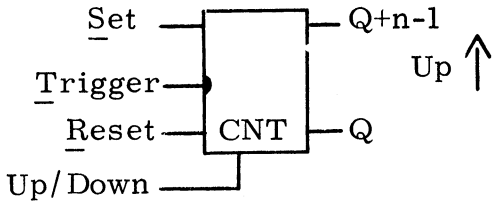
OR REGISTER



#, ORR, PD, N, I_1, ..., I_4

This parallel OR gate is used to "OR" together the consecutive outputs of several devices. Consider I, representing the first output of another variable length block. Successive inputs will come from successive outputs of the variable length block. LOGSIM creates N "OR" gates. (see example 6)

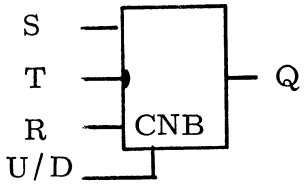
COUNTER Up/Down



#, CNT, PD, N, T, S, R, U/D

This N bit counter will count upwards if U/D is low on each change of the Trigger input from low to high. With U/D high the counter counts down. S and R obey the RS Flip-Flop truth table and applies to all levels of the counter at one time. The program creates N CNB blocks.

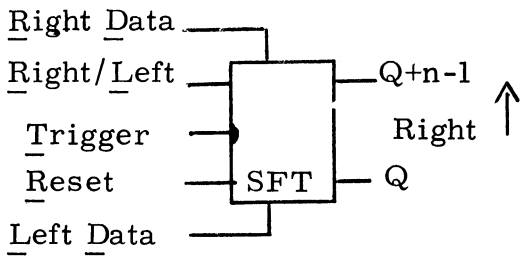
COUNTER BLOCK



#, CNB, PD, T, R, S, U/D

This block is generated by the computer when an N bit counter is defined. If the logic is useful for another purpose, it may be used by the user.

SHIFT REGISTER

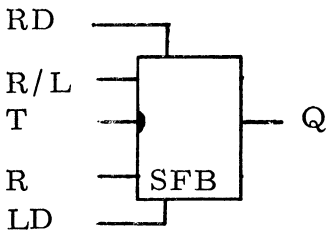


#, SFT, PD, N, T, RD, LD, R, R/L

This N level shift register will shift bits to the right when T changes from low to high provided R/L is low. If T changes from low to high and R/L is high, bits are shifted to the left. Bits to be shifted are entered by RD and bits to be shifted left are entered from LD. A change from low to high on the Reset will reset all N bits of the register. The program generates N SFB blocks.

SHIFT BLOCK

#, SFB, PD,T,RD,LD,R,R/L

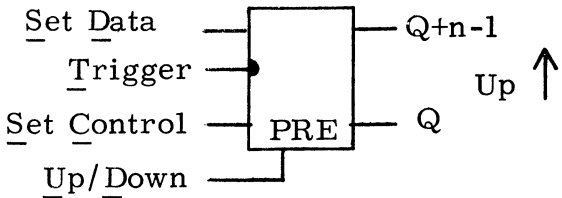


This block is generated by the computer when an N level shift register is defined. If the logic is useful for another purpose, it may be used by the user.

PRESET COUNTER

(UP/DOWN)

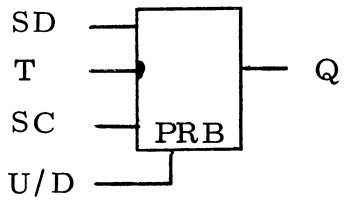
#, PRE,PD,N,T,SD,SC,U/D



This device is similar to the CNT counter except that the counter can be set in a different way. When SC is high, the counter will be set to the state of the N consecutive blocks beginning at the SD block. The program generates a PRB block.

PRESET COUNTER BLOCK

#, PRB,PD,T,SD,SC, U/D

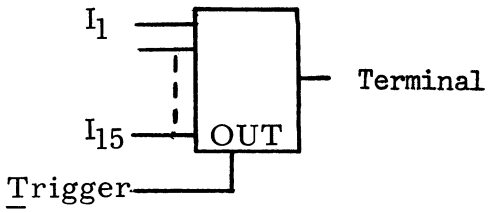


This block is generated by the computer when an N bit preset counter is defined. If the logic is useful for another purpose, it may be used by the user.

Special Blocks

The following blocks are special blocks used by LOGSIM to aid the user in the logic circuit simulation.

OUTPUT MODULE

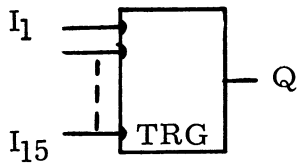


#, OUT, PD, T, I_1, I_2,...I_{15}

When T changes state, the states of I_1 through I_{15} will be printed at the terminal. The printing will occur PD time units after the trigger. The trigger may be from a Trigger Module as shown below.

Any combination of one thru fifteen inputs is legal.

TRIGGER MODULE

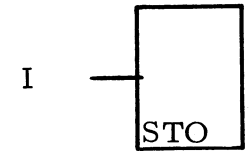


#, TRG, PD, I_1,..., I_{15}

Q will reverse state whenever any of the inputs I_1 through I_{15} changes state. This module is used primarily with the OUTPUT module.

Any combination of one thru fifteen inputs is legal.

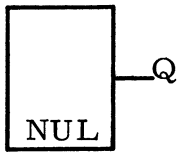
STOP



#, STO, PD, I

This block causes circuit simulation to cease PD time units after I goes high.

NULL



#, NUL, PD

This dummy block is used to replace an existing block in the circuit that is intended to be removed, but serves as an input to one or more other blocks. The logical value of the NUL is that value that causes it as an input to be ignored. If the block is not required at all, it may be removed entirely with the DELETE command.

VI. CIRCUIT OUTPUT

LOGSIM has two basic modes of execution, RUN and TRACE. A RUN mode simulation lists the states of selected blocks and the associated time to the terminal. A TRACE mode simulation lists the state and associated time of every logical state change. Both modes may be executed under the CYCLE option.

A simulation executed under the RUN mode requires that some output be requested by the user to be listed to the terminal.

OUT BLOCK

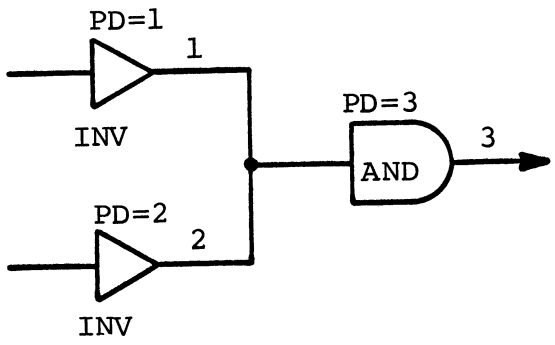
Output to be listed can be requested in one of two ways; by use of an OUT block in the circuit configuration or by a user output requested at run time.

An OUT block, described in Section V, requires as its inputs a list of up to 15 block names. The state of each input block name is printed at the terminal PD time units after the output trigger changes logical state. Output will be printed to the terminal at:

1. The beginning of each simulation.
2. The end of each simulation.
3. PD time units after an OUT block trigger is detected.

The propagation delay PD is useful when a user wishes to see the states of several blocks some settling time after a device has changed state.

Consider the following circuit:



Three Element Circuit

The performance of the circuit is meant to be instructional rather than useful. It has the static truth table shown below.

I_1	I_2	Q
F	F	T
F	T	F
T	F	F
T	T	F

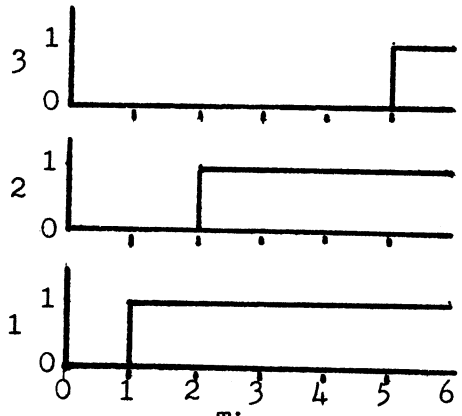
Introduce propagation delay and the circuit can take several routes to reach the same static state. If the block numbers also serve as their respective propagation delays, then we have a simple problem for analysis.

In general, the user must consider several things in analyzing a circuit:

1. Initial state of circuit (also see Section IV)
2. Method of displaying output
3. Method of defining inputs

1. Initial State of Circuit

LOGSIM will assume the logical state of false for every circuit element at time zero unless the user specifies otherwise. Assume that inputs I_1 and I_2 are both false. Then circuit will follow the timing diagram shown on the next page if all elements are false at time = 0.



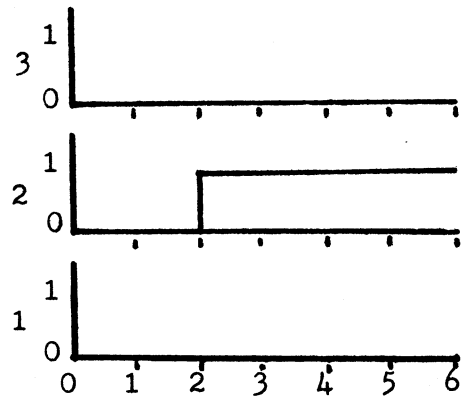
Timing Diagram

LOGSIM would produce the following output which is interpreted the same as the timing diagram as viewed from the left and has lines connecting the state markers.

```
BLOCK    1    2    3
NAME
  TIME  01   01   01
     0S  0    0    0
     1    \    \    \
     2     1    1    0
     5F     \    \    \
     5  STOP AT STEADY STATE
```

```
1,INV,1,F
2,INV,2,F
3,AND,3,1,2
```

However, if the input to block 1 were true and the input to block 2 false at time = 0, then the following diagram applies.



Timing Diagram

Similarly LOGSIM would have the following output.

```
BLOCK    1    2    3
NAME
  TIME  01   01   01
     0S  0    0    0
     2F  0    1    0
     2  STOP AT STEADY STATE
```

```
1,INV,1,T
2,INV,2,F
3,AND,2,1,2
```

Other ouputs result for other combinations of initial conditions. It is important that one realize the importance of initial conditions. In most circuits, however, an initial steady state is reached at which time a switch is activated and normal circuit action progresses.

2. Method of Displaying Output

As previously mentioned, the user may define an OUT block to select particular outputs or may define the outputs at run time. Both methods can result in the same output.

OUT BLOCK

If an OUT block is selected, it must appear in the circuit configuration. Only the first OUT block in the circuit configuration is used. Several OUT blocks may be specified in the circuit and used progressively by deleting the first that appears in the listing. An example of the use of an OUT block would be as follows. Every change in the circuit elements would like to be observed. This could be accomplished with the following two blocks.

```
4, OUT, 0, 5, 1, 2, 3

5, TRG, 0, 1, 2, 3
```

Block #4 is an output block with 0 propagation delay; it is triggered by any change in the state of block #5 and will display the outputs of blocks #1, #2, and #3.

Block #5 is the trigger block for block #4. With a 0 propagation delay, it will immediately change state and trigger #4 any time there is a change of state in blocks #1, #2, or #3.

This combination will produce the required output.

RUN TIME OUTPUT

If LOGSIM were instructed to run without the OUT and TRG blocks, the same output could be obtained at run time as follows.

```
:RUN
START TIME = 0
FINISH TIME = 10

BLOCKS TO TRIGGER OUTPUT = 1,2,3
BLOCKS TO BE OUTPUT = 1,2,3
OUTPUT DELAY = 0
```

3. Method of Defining Inputs

It is often useful to analyze a circuit with a variety of input combinations. This can be accomplished by the use of the CYCLE command. The CYCLE command allows the user to cycle up to 5 different switch registers over all possible combinations of true and false. With the switch registers used as inputs, it is possible to get 32 (2^5) different combinations of inputs during one simulation run.

Following is the simple circuit previously discussed and will be simulated with two switch register inputs cycled over all four possible combinations.

CYCLE Command Example

```
-LOGSIM
```

User types LOGSIM in the Tymshare EXECUTIVE.

```
:INPUT T

BLK DES: 1,INV,1,*1
BLK DES: 2,INV,2,*2
BLK DES: 3,AND,3,1,2
BLK DES: 4,OUT,0,5,1,2,3
BLK DES: 5,TRG,0,1,2,3
BLK DES: END
```

User then types INPUT T (T for terminal) and enters description of the circuit from the terminal. END returns LOGSIM to the command mode.

```
:LIST ALL

1       INV         1   *1
2       INV         2   *2
3       AND         3   1       2
4       OUT         0   5       1       2       3
5       TRG         0   1       2       3
```

```
:CYCLE *1-*2
TRACE ? N
OK
START TIME = 0
FINISH TIME = 10

* 1=F
* 2=F
  BLOCK    1    2    3
  NAME
    TIME   01   01   01
       OS  0    0    0
       1    1   0    0
       2    1    1   0
       5F   1    1    1
       5   STOP AT STEADY STATE
```

CYCLE plus a valid start and finish time begins the execution.

Block 1 changes from false to true at time = 1, block 2 at time = 2, and block 3 at time = 5, which is the circuit steady state condition.

```
* 1=T
* 2=F
  BLOCK    1    2    3
  NAME
    TIME   01   01   01
       OS  0    0    0
       2F  0     1   0
       2   STOP AT STEADY STATE

* 1=F
* 2=T
  BLOCK    1    2    3
  NAME
    TIME   01   01   01
       OS  0    0    0
       1F   1   0    0
       1   STOP AT STEADY STATE

* 1=T
* 2=T
  BLOCK    1    2    3
  NAME
    TIME   01   01   01
       OS  0    0    0
       OF  0    0    0
       0   STOP AT STEADY STATE

:QUIT

-
```

Cycling through the three remaining combinations.

The program is exited with the QUIT command.

TRACE MODE

If the user requires only the logical changes through the simulation, then the TRACE mode could be invoked to simply list every circuit block change of state. A sample of the TRACE mode can be seen in Section VII.

VII EXAMPLES

These examples are used to illustrate the operating features and conventions of LOGSIM.

EXAMPLE 1. Modulus 10 Counter

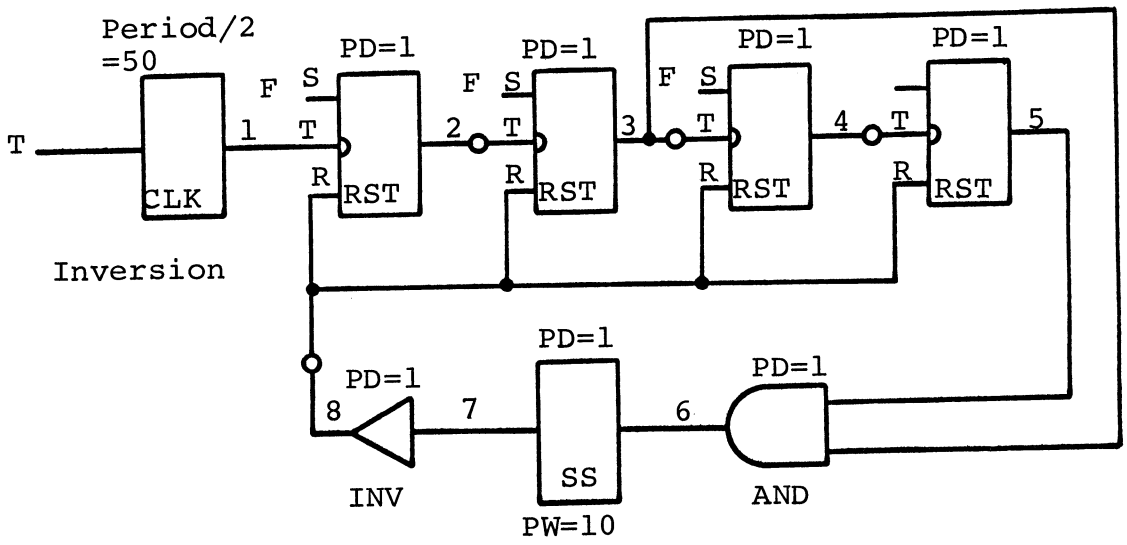


Figure 1 Modulus Counter

```
-EDITOR
*APPEND
1,CLK,50,T
2,RST,1,1,F,-8
3,RST,1,-2,F,-8
4,RST,1,-3,F,-8
5,RST,1,-4,F,-8
6,RST,----AND,1,3,5
7,SS,1,10,6
8,INV,1,7
9,OUT,20,1,1,2,3,4,5
*WRITE
TO RST
 NEW FILE
44 WORDS.
*QUIT
```

- The circuit description is entered in the Tymshare EDITOR. See EDITOR manual.

- A mistake is erased with $A^{c}A^{c}A^{c}A^{c}$ or the entire input line could be deleted with a Q^{c}.

- APPEND command is terminated with a control D.

- WRITE followed by a carriage return is typed and the contents of EDITOR written to a file named RST.

-LOGSIM

:INPUT RST

- This command instructs LOGSIM to read a file named RST.

:LIST ALL

- The block name ALL is a reserved block name to mean all blocks.

```
1        CLK      50   T
2        RST       1   1        F      -8
3        RST       1  -2        F      -8
4        RST       1  -3        F      -8
5        RST       1  -4        F      -8
6        AND       1   3        5
7        SS        1       10   6
8        INV       1   7
9        OUT      20   1        1      2      3      4      5
```

- A delay of 20 was used on the OUT block to allow time for the circuit to settle due to the propagation delays.

- An alternated way to list all blocks is as follows:

:LIST

BLK NAMES:

```
:RUN
START TIME = 0
FINISH TIME = 1500

  BLOCK    1    2    3    4    5
  NAME
    TIME  01   01   01   01   01
       0S 0    0    0    0    0
       70   1    1  0    0    0
      120  0     1  0    0    0
      170   1  0     1  0    0
      220  0    0     1  0    0
      270   1    1    1  0    0
      320  0     1    1  0    0
      370   1  0    0     1  0
      420  0    0    0     1  0
      470   1    1  0     1  0
      520  0     1  0     1  0
      570   1  0     1    1  0
      620  0    0     1    1  0
      670   1    1    1    1  0
      720  0     1    1    1  0
      770   1  0    0    0     1
      820  0    0    0    0     1
      870   1    1  0    0     1
      920  0     1  0    0     1
      970   1  0    0    0    0
     1020  0    0    0    0    0
     1070   1    1  0    0    0
     1120  0     1  0    0    0
     1170   1  0     1  0    0
     1220  0    0     1  0    0
     1270   1    1    1  0    0
     1320  0     1    1  0    0
     1370   1  0    0     1  0
     1420  0    0    0     1  0
     1470   1    1  0     1  0
     1500F 0     1  0     1  0
     1500   STOP AT TIME LIMIT
```

- Execution halts when the time limit is exceeded.

```
:TRACE
OK
START TIME = 600
FINISH TIME = 800

    TIME    BLK     TYP  STATE
     600    1       CLK    F
     650    1       CLK    T
     651    2       RST    T
     700    1       CLK    F
     750    1       CLK    T
     751    2       RST    F
     752    3       RST    F
     753    4       RST    F
     754    5       RST    T
     800    1       CLK    F
     800    STOP AT TIME LIMIT

:QUIT

-
```

- Circuit changes are monitored with the TRACE command over a portion of the previous time period.

- Notice that with the OUT block, the transients shown here are not observed in the previous output.

EXAMPLE 2. Octal Counter

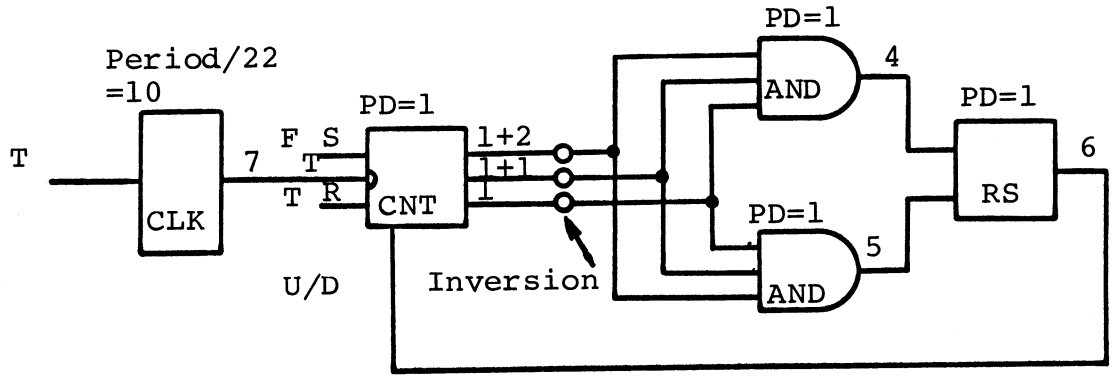


Figure 2 Octal Counter

The circuit will increment from 0 to 7 then decrement back to 0. Initially every device will be logical false. All devices have a delay of 1 unit except the clock which has a half period of 10 units.

```
-LOGSIM

:INPUT T                    - The circuit is described with
                              the input from T (terminal).

BLK DES: 1,CNT,1,3,7,F,F,6
BLK DES: 4,AND,1,1,1+1,1+2
BLK DES: 5,AND,1,-1,-1+1,-1+2
BLK DES: 6,RS,1,4,5
BLK DES: 7,CLK,10,T
BLK DES: END

:LIST

BLK NAMES: ALL

1       CNB      1  7      F      F      6
1  +1   CNB      1  1      F      F      6
1  +2   CNB      1  1  +1   F      F      6
4       AND      1  1      1  +1  1  +2
5       AND      1 -1     -1  +1 -1  +2
6       RS       1  4      5
7       CLK     10  T
```

```
:PRINT ALL

BLK NAME     TLAST     PRES     TPRED

1              0        U
1  +1          0        U
1  +2          0        U
4              0        U
5              0        U
6              0        U
7              0        U
```

- All block states are Undefined at this time. All blocks and switch registers not set by the user are set to logical false at run time.

```
:RESET ALL
OK
```

- The RESET command is used to show that all block initial states have been set to logical false by the user.

```
:PRINT ALL

BLK NAME     TLAST     PRES     TPRED

1              0        F
1  +1          0        F
1  +2          0        F
4              0        F
5              0        F
6              0        F
7              0        F
```

```
:RUN
START TIME = 0
FINISH TIME = 350

BLOCKS TO TRIGGER OUTPUT = 7
BLOCKS TO BE OUTPUT = 1,1+1,1+2,4,5,6,7
OUTPUT DELAY = 6

  BLOCK    1    1    1    4    5    6    7
  NAME          +1   +2
    TIME  01   01   01   01   01   01   01
       0S 0    0    0    0    0    0    0
       16 0    0    0    0     1   0     1
       26   1  0    0    0    0    0    0
       36   1  0    0    0    0    0     1
       46 0     1  0    0    0    0    0
       56 0     1  0    0    0    0     1
       66   1   1  0    0    0    0    0
       76   1   1  0    0    0    0     1
       86 0    0     1  0    0    0    0
       96 0    0     1  0    0    0     1
      106   1  0     1  0    0    0    0
      116   1  0     1  0    0    0     1
      126 0     1    1  0    0    0    0
      136 0     1    1  0    0    0     1
      146   1   1    1    1  0     1   0
      156 0     1    1  0    0     1   1
      166 0     1    1  0    0     1   0
      176   1  0     1  0    0     1   1
      186   1  0     1  0    0     1   0
      196 0    0     1  0    0     1    1
      206 0    0     1  0    0     1   0
      216   1   1   0    0    0     1   1
      226   1   1   0    0    0     1   0
      236 0     1   0    0    0     1    1
      246 0     1   0    0    0     1   0
      256   1  0    0    0    0     1    1
      266   1  0    0    0    0     1   0
      276 0    0    0    0     1   0     1
      286   1  0    0    0    0    0    0
      296   1  0    0    0    0    0     1
      306 0     1  0    0    0    0    0
      316 0     1  0    0    0    0     1
      326   1   1  0    0    0    0    0
      336   1   1  0    0    0    0     1
      346 0    0     1  0    0    0    0
      350F 0   0     1  0    0    0     1
      350  STOP AT TIME LIMIT
```

- The counter will require about 270-280 time units up to 7 then down to 0. A finish time of 350 is given to allow a margin to complete one cycle.

- All elements selected as output to be displayed 6 time units after each clock pulse.

- Counter is at 7.

- Counter is now 0.

- Counter starting up again.

```
:TRACE                                        - A TRACE is executed over
OK                                              an intermediate time range
START TIME = 120
FINISH TIME = 290

    TIME    BLK     TYP  STATE
     120    7       CLK    F
     121    1       CNB    F
     122    1  +1   CNB    T
     130    7       CLK    T
     140    7       CLK    F
     141    1       CNB    T
     142    4       AND    T        - Reset pulse of block 6 begins
     143    6       RS     T        - Counter is now sent to count down.
     150    7       CLK    T
     151    1       CNB    F
     152    4       AND    F
     160    7       CLK    F
     170    7       CLK    T
     171    1       CNB    T
     172    1  +1   CNB    F
     172    4       AND    T        - Reset pulse ends.
     173    4       AND    F
     180    7       CLK    F
     190    7       CLK    T
     191    1       CNB    F
     200    7       CLK    F
     210    7       CLK    T
     211    1       CNB    T
     212    1  +1   CNB    T
     213    1  +2   CNB    F
     213    4       AND    T
     214    4       AND    F
     220    7       CLK    F
     230    7       CLK    T
     231    1       CNB    F
     240    7       CLK    F
     250    7       CLK    T
     251    1       CNB    T
     252    1  +1   CNB    F
     260    7       CLK    F
     270    7       CLK    T
     271    1       CNB    F
     272    5       AND    T        - Set pulse of block 6 begins.
     273    6       RS     F        - Counter now is set to count down.
     280    7       CLK    F
     281    1       CNB    T
     282    5       AND    F        - Set pulse ends.
     290    7       CLK    T
     290    STOP AT TIME LIMIT
```

```
:PRINT ALL
```

- The logical states following the TRACE run are examined.

```
BLK NAME     TLAST     PRES     TPRED

1              281      T      8388607
1   +1         252      F      8388607
1   +2         213      F      8388607
4              214      F      8388607
5              282      F      8388607
6              273      F      8388607
7              290      T          300
```

```
:SAVE  UPDOWN
 OLD FILE
OK
```

- Present configuration is saved on a file named UPDOWN.

```
:QUIT
```

- UPDOWN is examined in the Tymshare EXECUTIVE mode.

```
-COPY  UPDOWN  TO T
```

```
1      ,CNB,        1, 7       , F       , F       , 6
1   +1 ,CNB,        1, 1       , F       , F       , 6
1   +2 ,CNB,        1, 1   +1  , F       , F       , 6
4      ,AND,        1, 1       , 1   +1  , 1   +2
5      ,AND,        1,-1       ,-1   +1  ,-1   +2
6      ,RS ,        1, 4       , 5
7      ,CLK,       10, T
```

```
-
```

- The size of this file can be reduced by deleting all blanks with the EDITOR.

This can be done as follows:

```
*SUBSTITUTE
"D^C" FOR "bD^C"
WAIT? N
```

where b=1 space

EXAMPLE 3. Binary Adder

The following circuit is used to add two octal words from a serial word generator.

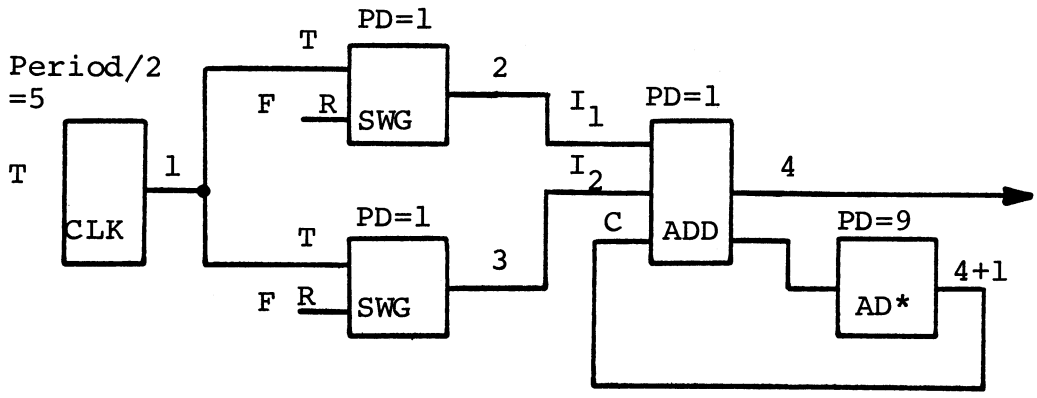


Figure 3 Binary Adder

```
-LOGSIM

:INPUT T

BLK DES: 1,CLK,5,T
BLK DES: 2,SWG,1,1,F,1
BLK DES: 3,SWG,1,1,F,23
BLK DES: 3,SWG,↑  4,ADD,1,2,3,4+1
BLK DES: 4-5,OUT,0,6,1,2,3,4+1,4
BLK DES: 6,TRG,0,1,2,3,4,4+1
BLK DES: END
```

- Note all eight digits are not required for the octal word in the SWG.
- A Q^C printed the ↑, which deleted the entire line. A^C deleted the 4. Manual writers make mistakes too.

```
:LIST ALL

1        CLK        5  T
2        SWG        1  1        F      00000001
3        SWG        1  1        F      00000023
4        ADD        1  2        3       4  +1
4  +1    AD*        1
5        OUT        0  6        1       2        3        4  +1    4
6        TRG        0  1        2       3        4        4  +1
```

- Note full eight digits shown for octal word.

```
:ALTER 4+1

BLK DES: 4+1,AD*,9
BLK DES: END
```

- The AD* is altered to increase the delay time in the feedback loop to the carry input of the ADD block.

```
:LIST 4-5

4        ADD          1  2        3        4  +1
4  +1    AD*          9
5        OUT          0  6        1        2        3        4  +1    4
```

- LIST is used only for the range of interest.

```
:RU
 ?
```

- An error in the command name.

```
:READ ALL
```

- Examining the SWG octal words.

```
2        SWG          1  1        F        00000001
PRESENT OCTAL WORD = 00000001
3        SWG          1  1        F        00000023
PRESENT OCTAL WORD = 00000023
```

With the octal words loaded in the SWG memory, the following binary addition will be performed.

```
                                        000 001
                                        010 011
                                        010 100
:RUN
START TIME = 0
FINISH TIME = 100

  BLOCK    1    2    3    4    4
  NAME                   +1
    TIME  01   01   01   01   01
       0S 0    0    0    0    0
        5   1  0    0    0    0
        6   1    1    1  0    0    - At time 6, 2 is T, 3 is T, carry is F,
       10  0     1    1  0    0      4 is F, with T carry.
       15   1    1    1    1  0
       16   1  0     1    1    1   - At time 16, 2 is F, 3 is T, carry is T,
       17   1  0     1    1  0      4 is F at time 17, with T carry.
       20  0   0     1    1  0
       25   1  0     1    1  0
       26   1  0    0     1  0    - At time 26, 2 is F, 3 is F, carry is T,
       27   1  0    0     1    1    4 is T at time 27, with no carry.
       30  0   0    0     1    1
       35   1  0    0    0     1
       36   1  0    0    0    0    - At time 36, 2 is F, 3 is F, carry is F,
       40  0   0    0    0    0      4 is F at time 37, with no carry.
       45   1  0    0    0    0
       46   1  0    1  0    0    - At time 46, 2 is F, 3 is T, carry is F,
       47   1  0    1  0    1      4 is T at time 47, with no carry.
       50  0   0    1  0    1
       55   1  0    1  0    1
       56   1  0    0    0    1
       57   1  0    0    0    0    - All F bits out of both SWG.
       60  0   0    0    0    0
       65   1  0    0    0    0
       66   1  0    0    0    0
       70  0   0    0    0    0
       75   1  0    0    0    0
       76   1  0    0    0    0
       80  0   0    0    0    0
       85   1  0    0    0    0
       86   1  0    0    0    0
       90  0   0    0    0    0
       95   1  0    0    0    0
       96   1  0    0    0    0
      100F 0   0    0    0    0
      100  STOP AT TIME LIMIT
```

- The SWG octal words are examined again. Note that all of the remaining bits in each SWG are 0. The original octal word was shifted out.

```
:READ ALL

2       SWG          1  1        F       00000001
PRESENT OCTAL WORD = 00000000
3       SWG          1  1        F       00000023
PRESENT OCTAL WORD = 00000000
```

```
:PRINT1--+- 1-4+1
```

- All commands followed by block names require one blank space separating the command and the block names.

```
BLK NAME     TLAST     PRES     TPRED

1              100       F         105
2               96       F     8388607
3               96       F     8388607
4               57       F     8388607
4  +1           35       F     8388607
```

```
:QUIT
```

- Time 8388607 is an infinite time to LOGSIM.

```
-
```

EXAMPLE 4. Using Shift Register

The following circuit is used to show the use of a shift register.

The register must be loaded with bits either by setting the blocks or entering the bits through RD or LD. In this example, simulation will be attempted without loading the register, and the effects of trying to change an initial condition on a block after one or more simulation periods have been executed will be shown.

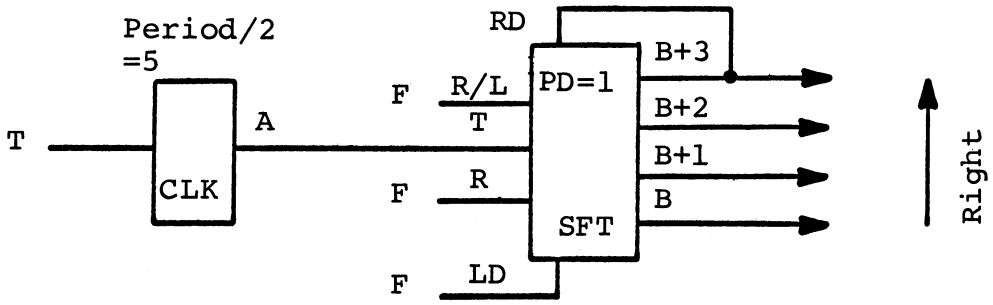


Figure 4. Right Shift Register

-LOGSIM

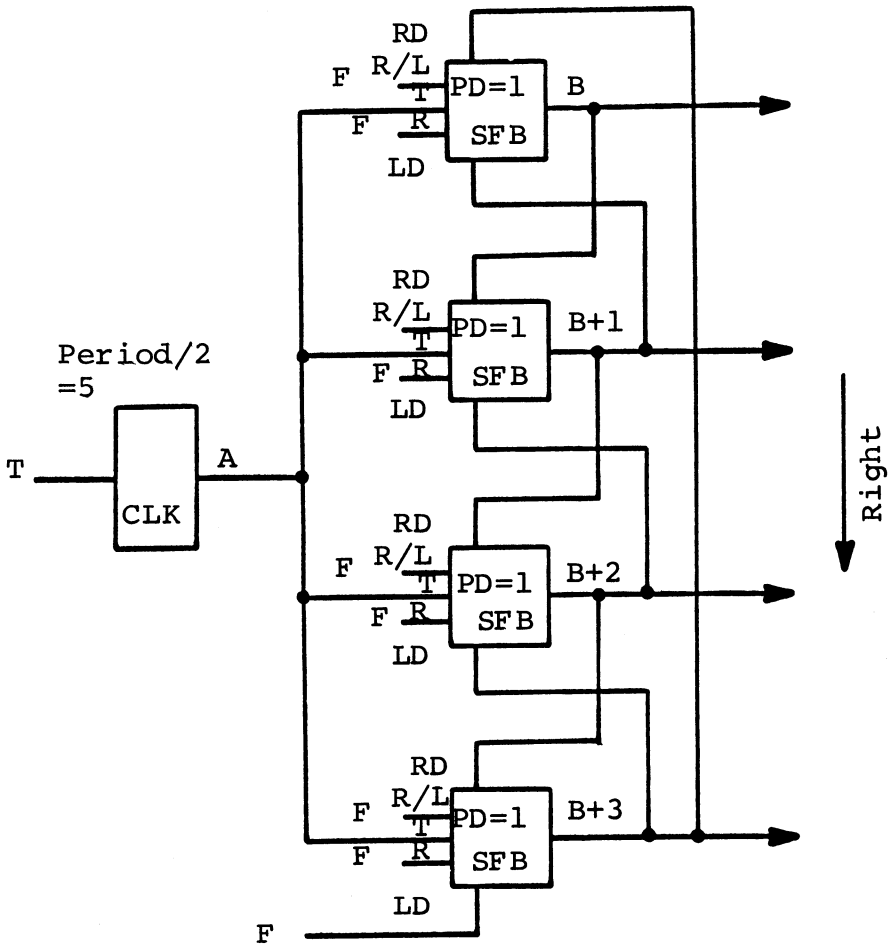
:INPUT T

BLK DES: A,CLK,5,T
BLK DES: B,SFT,1,4,A,B+3,F,F,F
BLK DES: END

```
*LIST ALL

A          CLK        5  T
B          SFB        1  A        B  +3    B  +1    F       F
B  +1      SFB        1  A        B        B  +2    F       F
B  +2      SFB        1  A        B  +1    B  +3    F       F
B  +3      SFB        1  A        B  +2    F        F       F
```

Redrawing the generated circuit.



```
:TRACE
OK
START TIME = 0
FINISH TIME = 50

    TIME    BLK     TYP  STATE
       5    A       CLK    T
      10    A       CLK    F
      15    A       CLK    T
      20    A       CLK    F
      25    A       CLK    T
      30    A       CLK    F
      35    A       CLK    T
      40    A       CLK    F
      45    A       CLK    T
      50    A       CLK    F
      50    STOP AT TIME LIMIT
```

- Since all of the registers were false, no change in the SFB blocks will occur.

```
:SET B
OK
```

- Block B is set to logical true.

```
:TRACE
OK
START TIME = 0
FINISH TIME = 50

    TIME    BLK     TYP  STATE
       5    A       CLK    T
      10    A       CLK    F
      15    A       CLK    T
      20    A       CLK    F
      25    A       CLK    T
      30    A       CLK    F
      35    A       CLK    T
      40    A       CLK    F
      45    A       CLK    T
      50    A       CLK    F
      50    STOP AT TIME LIMIT
```

- Note this run is the same as above. The SET command did not set block B. See the comments on initial conditions at the end of Section IV. In this case, the second start at time zero reinitialized the circuit to the initial conditions present before the first TRACE command.

A method of correcting this problem is outlined on the next page.

```
:INITIALIZE
OK
```

- Using the INITIALIZE command indicates to LOGSIM that new initial conditions must be assumed.

```
:SET B
OK
```

The purpose of this is to allow the user to examine the logical state of the circuit at the finish time of any run, then continue the run without recalculating the previous time period starting from zero.

```
:TRACE
OK
START TIME = 0
FINISH TIME = 75

   TIME   BLK     TYP  STATE
      5   A       CLK    T
      6   B       SFB    F
      6   B  +1   SFB    T
     10   A       CLK    F
     15   A       CLK    T
     16   B  +1   SFB    F
     16   B  +2   SFB    T
     20   A       CLK    F
     25   A       CLK    T
     26   B  +2   SFB    F
     26   B  +3   SFB    T
     30   A       CLK    F
     35   A       CLK    T
     36   B       SFB    T
     36   B  +3   SFB    F
     40   A       CLK    F
     45   A       CLK    T
     46   B       SFB    F
     46   B  +1   SFB    T
     50   A       CLK    F
     55   A       CLK    T
     56   B  +1   SFB    F
     56   B  +2   SFB    T
     60   A       CLK    F
     65   A       CLK    T
     66   B  +2   SFB    F
     66   B  +3   SFB    T
     70   A       CLK    F
     75   A       CLK    T
     75   STOP AT TIME LIMIT
```

B was set to T. After the first change of the clock from low to high, B shifted the true bit into B+1.

The T bit is shifted through all registers until it cycles back to block B.

```
:QUIT
```

EXAMPLE 5. Register Circuit

The following circuit is used to illustrate the interconnection of variable length blocks. Note in the circuit description on the next page, only the first two inputs for the gate (block C) and the OR register (block D) were specified. The consecutive inputs were taken as blocks directly following the first input specified.

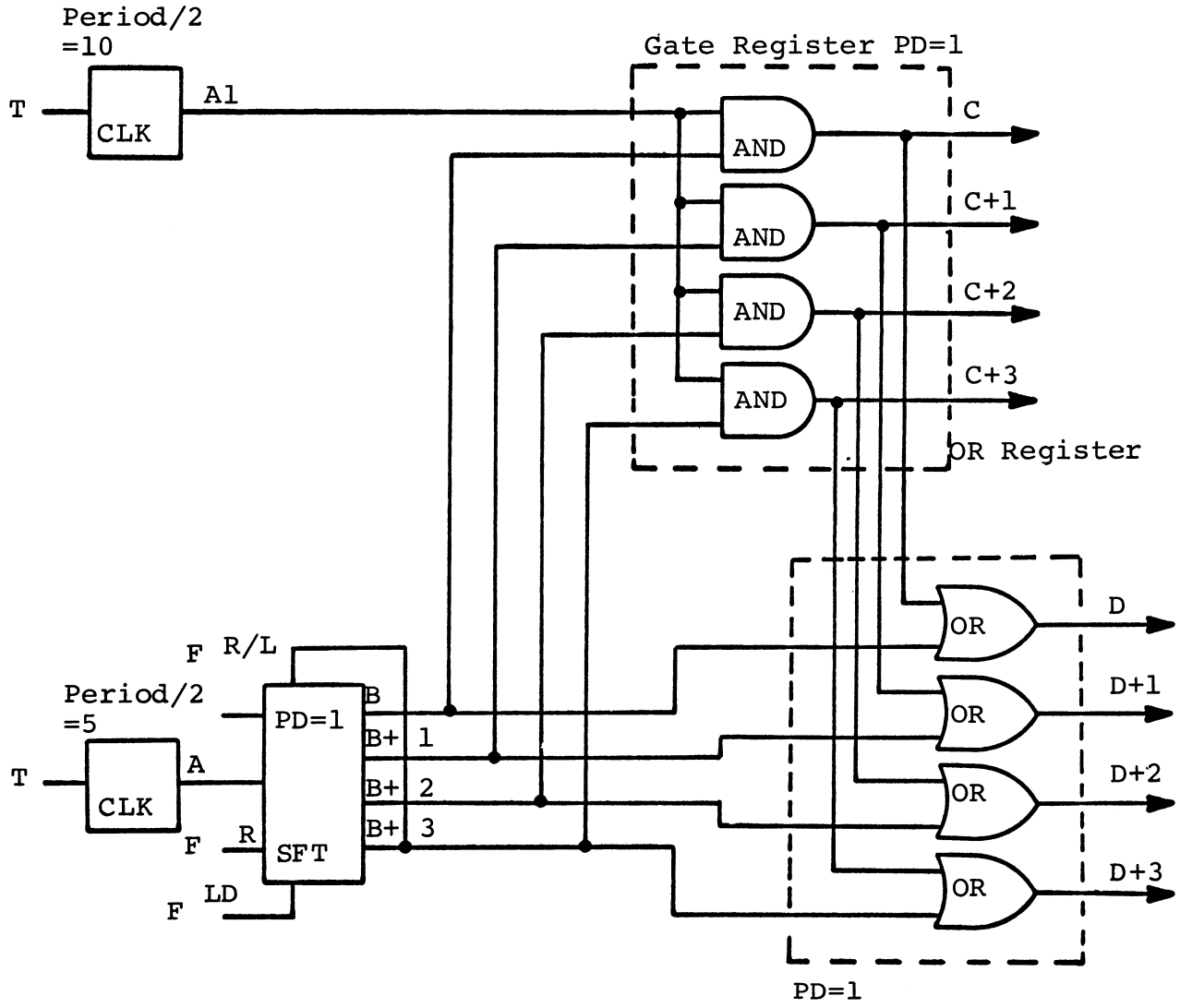


Figure 6. Register Circuit

```
-LOGSIM


:INPUT T


BLK DES: A,CLK,5,T
BLK DES: A1,CLK,10,T
BLK DES: B,SFT,1,4,A,B+3,F,F,F
BLK DES: C,GAT,1,4,B,A1
BLK DES: D,OBR,1,4,B,C
BLK DES: E,OUT,0,F1,A,A1,B,B+1,B+2,B+3,C,C+1,C+2,C+3,↵
D,D+1,D+2,D+3
BLK DES: F1,CLK,1,T
BLK DES: END                    - Note a line feed (↵) was typed after
                                  the "C+3," to continue with the
                                  description on the next line.

:LIST ALL

A       CLK      5  T
A1      CLK     10  T
B       SFB      1  A       B  +3   B  +1   F       F
B  +1   SFB      1  A       B       B  +2   F       F
B  +2   SFB      1  A       B  +1   B  +3   F       F
B  +3   SFB      1  A       B  +2   F       F       F
C       AND      1  B       A1
C  +1   AND      1  B  +1   A1
C  +2   AND      1  B  +2   A1
C  +3   AND      1  B  +3   A1
D       OR       1  B       C
D  +1   OR       1  B  +1   C  +1
D  +2   OR       1  B  +2   C  +2
D  +3   OR       1  B  +3   C  +3
E       OUT      0  F1      A       A1      B       B  +1   B  +2
                    B  +3   C       C  +1   C  +2   C  +3   D
                    D  +1   D  +2   D  +3
F1      CLK      1  T

                        The clock (block F1) with a half
                        period=2 was used to trigger the
                        OUT block to observe output at
                        every time unit.
```

```
:SET B
OK
```

Block B is set to true.

```
:RUN
START TIME = 0
```

The alpha character O was typed instead of a zero.

```
*** START TIME NOT NUMERIC
START TIME= 0
FINISH TIME = 30
```

BLOCK NAME	A	A1	B	B +1	B +2	B +3	C	C +1	C +2	C +3	D	D +1	D +2	D +3
TIME	01	01	01	01	01	01	01	01	01	01	01	01	01	01
0S	0	0	1	0	0	0	0	0	0	0	0	0	0	0
1	0	0	1	0	0	0	0	0	0	0	1	0	0	0
2	0	0	1	0	0	0	0	0	0	0	1	0	0	0
3	0	0	1	0	0	0	0	0	0	0	1	0	0	0
4	0	0	1	0	0	0	0	0	0	0	1	0	0	0
5	1	0	1	0	0	0	0	0	0	0	1	0	0	0
6	1	0	0	1	0	0	0	0	0	0	1	0	0	0
7	1	0	0	1	0	0	0	0	0	0	0	1	0	0
8	1	0	0	1	0	0	0	0	0	0	0	1	0	0
9	1	0	0	1	0	0	0	0	0	0	0	1	0	0
10	0	1	0	1	0	0	0	0	0	0	0	1	0	0
11	0	1	0	1	0	0	0	1	0	0	0	1	0	0
12	0	1	0	1	0	0	0	1	0	0	0	1	0	0
13	0	1	0	1	0	0	0	1	0	0	0	1	0	0
14	0	1	0	1	0	0	0	1	0	0	0	1	0	0
15	1	1	0	1	0	0	0	1	0	0	0	1	0	0
16	1	1	0	0	1	0	0	1	0	0	0	1	0	0
17	1	1	0	0	1	0	0	0	1	0	0	1	1	0
18	1	1	0	0	1	0	0	0	1	0	0	0	1	0
19	1	1	0	0	1	0	0	0	1	0	0	0	1	0
20	0	0	0	0	1	0	0	0	1	0	0	0	1	0
21	0	0	0	0	1	0	0	0	0	0	0	0	1	0
22	0	0	0	0	1	0	0	0	0	0	0	0	1	0
23	0	0	0	0	1	0	0	0	0	0	0	0	1	0
24	0	0	0	0	1	0	0	0	0	0	0	0	1	0
25	1	0	0	0	1	0	0	0	0	0	0	0	1	0
26	1	0	0	0	0	1	0	0	0	0	0	0	1	0
27	1	0	0	0	0	1	0	0	0	0	0	0	0	1
28	1	0	0	0	0	1	0	0	0	0	0	0	0	1
29	1	0	0	0	0	1	0	0	0	0	0	0	0	1
30F	0	1	0	0	0	1	0	0	0	0	0	0	0	1
30	STOP AT TIME LIMIT													

```
:QUIT

-
```

EXAMPLE 6. Gate Register

The following circuit illustrates the conventions used by LOGSIM in defining input block names.

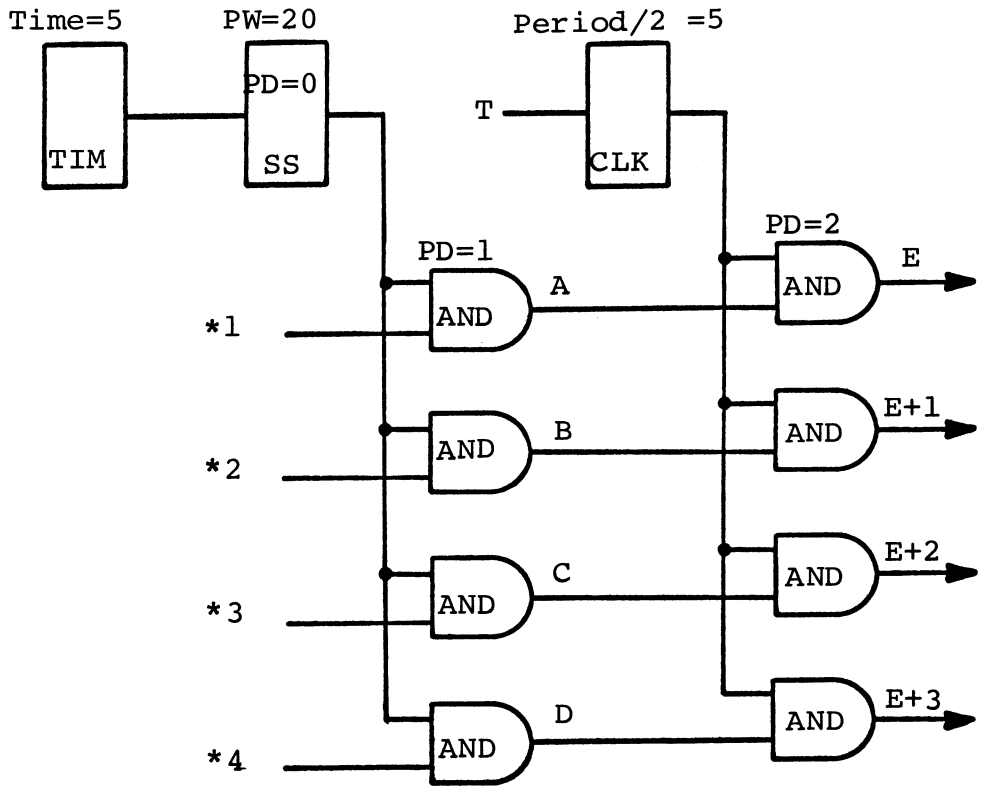


Figure 7. Gate Register Circuit

```
-LOGSIM

:INPUT T

BLK DES: TIME,TIME,5
BLK DES: SS,SS,0,20,TIME
BLK DES: A,AND,1,*1,SS
BLK DES: B,AND,1,*2,SS
BLK DES: C,AND,1,*3,SS
BLK DES: D,AND,1,*4,SS
BLK DES: E,GAT,2,4,A,CLK1
BLK DES: CLK1,CLK,5,T
BLK DES: OUT,OUT,0,CLK2,TIME,SS,A,B,C,D,CLK1,E,E+1,E+2,E+3
BLK DES: CLK2,CLK,1,T
BLK DES: END
```

- Note the spelling of TIME for the block type with the name TIME.

- Clock (CLK2) is used to trigger the OUT block at every time unit.

- Note here, the E was truncated from the block type without causing an error message.

```
:LIST ALL

TIME     TIM      5
SS       SS       0        20 TIME
A        AND      1  *1      SS
B        AND      1  *2      SS
C        AND      1  *3      SS
D        AND      1  *4      SS
E        AND      2  A       CLK1
E  +1    AND      2  A  +1   CLK1
E  +2    AND      2  A  +2   CLK1
E  +3    AND      2  A  +3   CLK1
CLK1     CLK      5  T
OUT      OUT      0  CLK2    TIME    SS      A       B       C
                     D       CLK1    E       E  +1   E  +2   E  +3
CLK2     CLK      1  T
```

- Note the inputs to the gate register block. There are no blocks with the names A+1, A+2, and A+3. Note the listing after the run period.

```
:SET *1,*3
OK
OK
```

Switch registers 1 and 3 are set to logical true. The OK indicates execution of the command for each block or block range seperated by a comma.

The first three characters of a block name are printed on the first line with the remaining three on the second line. Block names with (+) are considered two part names with the first three characters in the first part and the +NN as the second part.

```
:RUN
START TIME = 0
FINISH TIME = 30
```

BLOCK NAME / TIME	TIM E 01	SS 01	A 01	B 01	C 01	D 01	CLK 1 01	E 01	E +1 01	E +2 01	E +3 01
0S	0	0	0	0	0	0	0	0	0	0	0
1	0	0	0	0	0	0	0	0	0	0	0
2	0	0	0	0	0	0	0	0	0	0	0
3	0	0	0	0	0	0	0	0	0	0	0
4	0	0	0	0	0	0	0	0	0	0	0
5	1	1	0	0	0	0	1	0	0	0	0
6	1	1	1	0	1	0	1	0	0	0	0
7	1	1	1	0	1	0	1	0	0	0	0
8	1	1	1	0	1	0	1	1	0	1	0
9	1	1	1	0	1	0	1	1	0	1	0
10	1	1	1	0	1	0	0	1	0	1	0
11	1	1	1	0	1	0	0	1	0	1	0
12	1	1	1	0	1	0	0	0	0	0	0
13	1	1	1	0	1	0	0	0	0	0	0
14	1	1	1	0	1	0	0	0	0	0	0
15	1	1	1	0	1	0	1	0	0	0	0
16	1	1	1	0	1	0	1	0	0	0	0
17	1	1	1	0	1	0	1	1	0	1	0
18	1	1	1	0	1	0	1	1	0	1	0
19	1	1	1	0	1	0	1	1	0	1	0
20	1	1	1	0	1	0	0	1	0	1	0
21	1	1	1	0	1	0	0	1	0	1	0
22	1	1	1	0	1	0	0	0	0	0	0
23	1	1	1	0	1	0	0	0	0	0	0
24	1	1	1	0	1	0	0	0	0	0	0
25	1	0	1	0	1	0	1	0	0	0	0
26	1	0	0	0	0	0	1	0	0	0	0
27	1	0	0	0	0	0	1	0	0	0	0
28	1	0	0	0	0	0	1	0	0	0	0
29	1	0	0	0	0	0	1	0	0	0	0
30F	1	0	0	0	0	0	0	0	0	0	0

```
30  STOP AT TIME LIMIT
```

:LIST ALL

```
TIME     TIM        5
SS       SS         0          20 TIME
A        AND        1  *1        SS
B        AND        1  *2        SS
C        AND        1  *3        SS
D        AND        1  *4        SS
E        AND        2  A         CLK1
E  +1    AND        2  B         CLK1
E  +2    AND        2  C         CLK1
E  +3    AND        2  D         CLK1
CLK1     CLK        5  T
OUT      OUT        0  CLK2      TIME    SS      A       B       C
                       D         CLK1    E       E  +1   E  +2   E  +3
CLK2     CLK        1  T
```

Note that LOGSIM has assumed block A as the first input to block E, the second A+1=B, A+2=C, and A+3=D. See the following page for more details.

:QUIT

-

LOGSIM checks for complete block names with (+) for inputs. If none are found, it searches for the first part of the input block name (the first three characters) to match the characters in front of the (+) in the input block name.

Consider the case where the following block names are in an existing circuit in LOGSIM.

BLK NO.	BLK NAME
1	ABLE
2	BAKER
3	ABL+5
4	CHARLY
5	ABL1
6	ABL2

If the inputs were ABL1, ABL+1, ABL+4, and ABL+5, LOGSIM would make the following match.

Input Name	BLK NO.
ABL1	5
ABL+1	2
ABL+3	4
ABL+5	3

VIII. ERROR MESSAGES

COMMAND Errors

```
:ERROR1
 ?

:ERROR2
FOR ASSISTANCE TYPE 'HELP'



:INS
AMBIGUOUS COMMAND, TYPE MORE CHARACTERS

:
```

Errors in specifying the command will print the above error messages.

SPECIFICATION Errors

Specification error checking will occur when entering any circuit block description.

A. When input is from the terminal.

The circuit description is checked after each line is entered. If an error is found in the block description, an error message is printed on the terminal and the block description deleted.

A new block description may be entered directly after the error message or entered at the completion of the INPUT mode with the ADD or INSERT command.

B. When input is from a file.

Error checking will occur after reading each line from a file. If the block description is incorrect, that line will be deleted from the circuit and the description printed along with the error message.

Corrections after reading the file can be made by:

1. Correcting the error in the file and reentering the file again with the INPUT command.

2. Reentering the flagged blocks with the correct specifications with the ADD or INSERT command.

If the circuit has all correct block specifications but is logically wrong, corrections can be made with the ALTER, DELETE, and INSERT commands.

Block Description Error Messages

1. #, TYPE, PD, I_1, ..., I_n

```
BLK DES: A,AN

*** BLOCK DESCRIPTION NOT COMPLETE
```

2. #, TYPE, PD, I_1, ..., I_n

```
BLK DES: A,AND,F,T,F

*** PROGATION DELAY ERROR,  0<=PD<=8388607
```

.

3. #, TYPE, PD, I_1, ..., I_n

```
BLK DES: A,ANN,5,T,F

*** NOT A VALID BLOCK TYPE
```

4. #, TYPE, PD, I_1, ..., I_n

```
BLK DES: A,OR,4,C,D
BLK DES: A,AND,4,C,S

*** BLOCK NAME PREVIOUSLY USED
```

5. #, TYPE, PD, I_1, ..., I_n

```
BLK DES: C,INV,5

*** TYPE REQUIRES A DIFFERENT NUMBER OF PARAMETERS
```

6. #, TYPE, PD, I_1, ..., I_n

```
BLK DES: D,ORR,5,F,I,G

*** F
*** PARAMETER NOT NUMERIC
```

7. #, TYPE, PD, I_1, ..., I_n

```
BLK DES: F,SWG,O,T,R,7778

*** ILLEGAL OCTAL WORD IN SWG
```

The octal word is limited to 8 digits for 24 bits with one digit for each three bits.

8. #, TYPE, PD, I_1, ..., I_n

```
BLK DES: A,CNT,4,5,I,I,I,I
BLK DES: A+1,OR,5,6

*** BLOCK NAME PREVIOUSLY USED
```

Block name A generated CNB blocks with names A, A+1, A+2, A+3, and A+4.

9. #, TYPE, PD, I_1, ..., I_n

BLK DES: B+1,AND,5,6
BLK DES: B,ORR,2,5,4,5

***** GENERATED BLOCK NAME USED**

Block name B+1 is a block name generated for the Or register.

10. #, TYPE, PD, I_1, ..., I_n

BLK DES: A,GAT,4,35,I,G

***** GENERATED BLOCKS EXCEED TYPE LIMIT**

The maximum number of blocks that can be generated with a variable length block is 33.

11. #, TYPE, PD, I_1, ..., I_n

BLK DES: C,INV,5,*37

***** SWITCH NUMBER >36**

Switch registers may be -*1 thru -*36 and *1 thru *36.

12. #, ADD, PD, I1, I2, C

BLK DES: A,ADD,4,I1,I2,C

***** CARRY BLOCK NAME PREVIOUSLY USED**

The carry block name #+1 was previously used.

Block Name Error Messages

13. Bl, B2, ...Bn ALL or $

```
:LIST A

*** INVALID BLOCK NAME DESCRIPTION
```

Bock A is not in the existing circuit. Switch register names are not legal for the commands LIST, DELETE, ALTER, and INSERT.

14. Bl, B2, ...Bn, ALL or $

```
:PRINT 1-*2

*** RANGE CAN NOT INCLUDE BLOCK AND SWITCH
```

Block and switch register names can not be included in an inclusive range. Switch and block names are legal as consecutive or grouped types for PRINT, SET, and RESET. A :PRINT A-D, *3, *10-*30, Z would be a legal command if block names A inclusive to D and Z were existing circuit blocks.

15. Bl, B2, ...Bn, ALL or $

```
:PRINT 1,2,B

NAME          TLAST     PRES     TPRED

1                 0      U
2                 0      U
*** INVALID BLOCK OR SWITCH DESCRIPTION
```

Blocks 1 and 2 were existing blocks but block B was not.

CYCLE Command Error Messages

16. B1, B2, ...Bn (only switch numbers)

```
:CYCLE A
*** SWITCH NUMBER NOT NUMERIC
```

```
:CYCLE
SWITCH NUMBER: *40
*** ILLEGAL SWITCH NUMBER
```

Legal switch register numbers are 1 thru 36.

17. B1, B2, ...Bn (only switch numbers)

```
:CYCLE *1-*6
*** A MAXIMUM OF 5 SWITCHES MAY BE CYCLED
```

Switch numbers may be with or without *.

18. COMMAND file name

:DUMP T

***** DUMP OR RECOVER COMMANDS CAN NOT HAVE THE TERMINAL AS A FILE NAME**

DUMP of the present circuit status
can only be to a file.

19. COMMAND file name

:RECOVER SWG

***** RECOVER FILE NOT CREATED BY DUMP COMMAND**

The RECOVER command is only valid for
files created with the DUMP command.

Run Time Error Messages

20. No Circuit Present

```
:RUN
*** NO CIRCUIT
```

21. Start and Finish Time Error Messages

```
:RUN
START TIME = A

*** START TIME NOT NUMERIC

START TIME= -5

*** TIME < ZERO


:RUN
START TIME = 10
FINISH TIME = 5

*** FINISH TIME LESS THAN START TIME


START TIME = 5
FINISH TIME = -5

*** TIME < ZERO
```

22. Undefined Inputs

```
BLK DES: A,AND,5,B,C
BLK DES: END

:RUN
START TIME = 0
*** INPUT  B      FOR BLOCK NAME A      IS UNDEFINED
*** INPUT  C      FOR BLOCK NAME A      IS UNDEFINED

CONTINUE RUN WITH UNDEFINED INPUTS ASSUMED LOGICAL FALSE? Y
FINISH TIME = ⊕,
```

Execution was stopped with an ALTMODE and a carriage return.

```
:LIST ALL

A       AND        5  F       F
```

Note: Undefined inputs are changed to logical false.

23. Block Input Error Messages During Simulation

```
A,RS,0,T,T
```

```
*** R AND S OF BLOCK A       ARE BOTH ON AT TIME=          0
```

Block A as shown has both Set and Reset inputs on at the same time.

```
A,THR,0,B,T,T,T,T
B,AND,5,T
```

```
*** BLOCK NAME B        IS NOT A THRESHOLD LEVEL FOR BLOCK A
```

The THL# block was an AND gate instead of a THL type block.

```
A,THR,0,B,T,T,T,T
B,THL,100,10,20
```

```
*** MISMATCH IN NUMBER OF WEIGHTS FOR THRESHOLD BLOCK A
```

There were only two weights specified in block B and there were four inputs for block A.

TYMSHARE, INC.

525 UNIVERSITY AVENUE, SUITE 220
PALO ALTO, CALIFORNIA