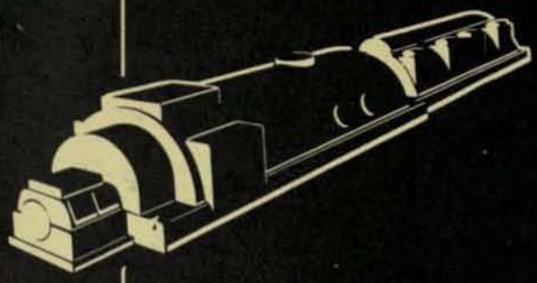

March 10, 1959 DF-59LS23

LOGTAB - A LOGIC TABLE TECHNIQUE

J. KING

LARGE STEAM TURBINE-GENERATOR
DEPARTMENT

GENERAL ELECTRIC

SCHENECTADY, N. Y.

TG-1013 (1-52)

FOR USE OF G-E EMPLOYEES ONLY

GENERAL ELECTRIC

TECHNICAL INFORMATION SERIES

Title Page

AUTHOR	SUBJECT CLASSIFICATION	NO.
Jane E. King	COMPUTERS, AUTOMATIC PROGRAMMING.	DF-59LS23
		DATE March 10, 1959

TITLE LOGTAB - A LOGIC TABLE TECHNIQUE

ABSTRACT LOGTAB is a method for expressing the logic of a computer program in table form and the subsequent interpretation of the table by the computer. The external and internal language and the method of interpretation on the IBM 704 are described. LOGTAB is an extension of TABSOL for the IBM 702.

G.E. CLASS	REPRODUCIBLE COPY FILED AT	NO. PAGES
3	Tech. Report Library, LST-G Dept., Bldg. 273, room 319, Schenectady, New York	11
GOV. CLASS. -		

CONCLUSIONS LOGTAB appears to be shorter in length and faster in execution than other methods of table look-up for multiple arguments on a computer. These criteria have not been fully explored for program logic but it has the advantage of requiring no programming once a table has been formed. In other words, the table replaces the flow chart and hand coding of a program and remains as a difinitive and understandable record of the program.

By cutting out this rectangle and folding on the center line, the above information can be fitted into a standard card file.

For list of contents—drawings, photos, etc. and for distribution see next page

INFORMATION PREPARED FOR LST-G DEPT.

TESTS MADE BY

COUNTERSIGNED H. N. CANTRELL Harry N. Cantrell DIV. MANAGER, COMPUTATIONAL RESEARCH

DIVISIONS LST-G ENGINEERING LOCATION SCHENECTADY, New York

13-A (1-52)

GENERAL ELECTRIC COMPANY
TECHNICAL INFORMATION SERIES
CONTENTS PAGE

CONTENTS OF REPORT DF-59LS23

NO. PAGES TEXT 7

NO. CHARTS 4

DRAWING NOS.

PHOTO NOS.

DISTRIBUTION (15) copies of report to LST-G Library, Bldg. 273-319
(52) copies of title page to LST-G Library, Bldg. 273-319

SCHENECTADY

RE Callaghan	59-214	HN Cantrell	59-244		
JL Ogle	273-2070	A Sturges	273-2070		
RE May	273-2070	JE King (2)	59-214		
DB ManMillan	Bldg. E-6, room 220A KAPL			DM Bouton	Bldg. 32, room 302
PM Currier	Bldg. 5, room 523			HW Penske	Bldg. 18-A, room 400
DF Langenwalter	Bldg. 37, room 578			JW Pontius	Bldg. 5, room 517
AH Rau	Bldg. 36, room 631				

Charles W. Libby — Lynn Computations, Bldg. 59, River Works, West Lynn 3, Mass.
A Keller — Lynn River Works, Lynn, Mass.
B Grad — NEW YORK OFFICE, Manufacturing Services.
TF Kavanagh — NEW YORK OFFICE, Manufacturing Services.
Hayden E. Williams — Data Processing Operations, Electronics Div. GE Co. Syracuse, NY
RA Habermann — HMEE Dept., Court St., Syracuse, NY
Cortland Diehm — Missile-Ordance Systems Dept., 3198 Chestnut St. Phila., Pa.
SB Williams — Power Transformer Dept., Pittsfield, Mass. Bldg. 17-260
Dr. Robert S. McCready — Aircraft Nuclear Propulsion, Cincinnati 15, Ohio
DD Degler — T&I Bldg., Orange & Van Ness St., Tempe, Arizona
AI Benson — T&I Bldg., Orange & Van Ness St., Tempe, Arizona
HM Sassenfeld — T&I Bldg., Orange & Van Ness St., Tempe, Arizona
H Tellier — Data Processing, Hanford Atomic Products, Richland, Washington
GA Hagerty — 13430 North Black Canyon Highway, Phoenix, Arizona
CC Lasher — 13430 North Black Canyon Highway, Phoenix, Arizona
ES Spiegelthal — 7235 Wisconsin Avenue, Suite 201, Bethesda, Maryland
J Weizenbaum — 951 Commercial Street, Palo Alto, California
JA Porter — FPLD, Bldg. 305 - EVENDALE
JB Hansen — 7235 Wisconsin Avenue, Suite 201, Bethesda, Maryland

TABLE OF CONTENTS DF-59LS23

INTRODUCTION

The IBM 702 program TABSOL is a generalized interpretive table solver prepared by the Integrated Systems Project, Materials Service of General Electric in New York, New York. TABSOL was studied for possible application and execution as a 704 program and the resulting program is LOGTAB (LOGIC TABLES). The basic principles of TABSOL were preserved but extensions and modifications were made in an attempt to make it applicable to a larger class of problems and to facilitate implementation on the IBM 704. In the following description of LOGTAB, no attempt is made to distinguish where TABSOL leaves off and LOGTAB begins.

GENERAL FEATURES OF LOGTAB

In using LOGTAB, all logical decisions of a program are expressed in tables. Each table has four major divisions which we shall call quadrants A, B, C, and D and their positions are shown in Figure 1.

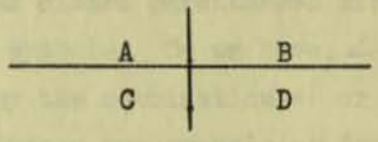

Figure 1

The contents of each quadrant is as follows:

A - references to all items to be tested in making the decisions included in table

B - The separate tests to be made on the items in A

C - references to items that are to be modified as a result of meeting the tests made on items in A

D - the specific things to be done to items in C as a result of tests in B on items in A

All information relating to an item in A or C is given in a row of the table and all information necessary for one "condition" (tests and results) are given in a column of the table. In addition, the table must be read from left to right and when a set of conditions (B) are met in one column, the results (D) for that column are executed and columns to the right are not tested. The schematic table (Figure 2) shows how this principle is expressed.

If and if	this not this	this this	(not tested)
then do and	(go to next column)	this this	

Figure 2

ALLOWABLE TABLE ENTRIES

In attempting to make the table prepared for the computer look as much like the table for human consumption as possible, some compromises in symbols and notation had to be made. However, as few changes and rules as possible were made. As an example, the less than ($\angle$) and greater than ($\succ$) symbols in every day use are not permissible computer characters. Therefore, they have been changed to open and closed parentheses since these allowable characters are most nearly like these symbols. So we have, $\angle$ = (and $\succ$ =). Similarly, the symbol $\leq$ is replaced by the combination =(or (=. Since it is impossible computer-wise to superimpose two symbols as for "not equal" ($\neq$) , this symbol has been replaced by () meaning less than or greater than. Another departure from "human" nomenclature is used to express the absolute value of a quantity. Vertical lines are not permissible computer characters so an absolute value is expressed with slashes.

Example: |AB| = /AB/.

The following forms of elements are permissible in the quadrants indicated:

	Element in Quadrant A	Meaning
1.	ABC ABC(2) ABC(N) ABC(N+1)	The quantity identified by the symbol (with its subscript, if given).

	Element in Quadrant A	Meaning
2.	* 10 *PQR	The quantity which can be found by computing a function identified by a symbol (PQR) or by statement number as in Fortran (10)
3.	(Blank)	This implies an "or" condition. It means that an additional test is to be made on the element identified in A above the blank or blanks.

A second column in quadrant A gives the condition for the test and the permissible tests are: (,),=,(=,=(,)=,=), and ( ).

	Element in Quadrant B	Meaning
1.	1.75 2	The test on the element in A is to be made against the exact value given.
2.	AB AB(2) AB(N) AB(N-1)	The test on the element in A is to be made against the quantity identified by the symbol (with its subscript, if given)
3.	* 10 * PQR	The test on the element in A is to be made against the quantity found by computing the function identified by a symbol (PQR) or by a statement number (10).
4.	-	Make no test on element in A.

	Element in Quadrant C	Meaning
1.	AB AB(2) AB(N) AB(N+1)	Replace the quantity identified by the symbol (with its subscript, if given) as indicated in D.
2.	+X	Increase X (where X is of the form in 1 above) by the amount indicated in D.
3.	-X	Decrease X by the amount indicated in D.
4.	(blank)	Execute the indicated operation in D.

	Element in Quadrant D	Meaning
1.	1.5 .375	Use exact value as indicated in C.
2.	ABD ABD(2) ABD(N) ABD(N-1)	Use the quantity identified by the symbol (with its subscript, if given) as indicated in C.
3.	* 10 * PQR	Compute function identified by symbol (PQR) or by statement number (10) and use as indicated in C.
4.	**XYZ **140	Go to function or table identified by symbol (XYZ) or by statement number (140).

Note that the symbol * essentially means "go to and come back here" and that ** means "go to unconditionally".

Figure 3 shows a sample LOGTAB table. Reading it in English, it says: If (the absolute value of the N^{th} element in the array AB) is less than 1.75, and if (the result of computing function called PQR) is less than or equal to (the result of computing function called AZ), and if the quantity IR2 is less than N; then, replace (the third element in the array SP) with (the result of computing the function called RS), increase the quantity IR2 by 1, compute function called DUZ, and go to function T16 (this may be another table). But if one of the previous conditions was not met, and if (the absolute value of the N^{th} element in the array AB) is less than 3.5, and if the quantity IR2 is less than N or if it is = N; then, increase the quantity IR2 by 2, decrease the quantity AX by 1.5 and go to the function or table T12. However, if none of the previous conditions were met, increase the quantity IR2 by 1, decrease the quantity AX by (minus N), and go to T12.

Note that all columns in quadrant D end with a ** (go to).

- idea of not switching columns—absolute value

No all other concept

/AB(N)/	(	1.75	3.5	-
*PQR	(=	*AZ	-	-
IR2	(	N	N	-
	=	-	N	-
SP(3)		*RS	-	-
+IR2		1	2	1
-AX		-	1.5	-N
		*DUZ	**T12	**T12
		**T16		

FIGURE 3

COMPUTER METHOD

Several methods of machine expression of the tables and machine interpretation of these expressions were studied. The following observations were made:

1. A given table, if stored in its entirety in the machine, required too much space because of the redundant nature of table elements.

2. A program to unravel the details of a complete table required too much time per entry, although the interpretation routine itself did not require much space.

3. If the information of the given table was packed into the computer to save space, the execution time for unpacking and testing each entry was excessive.

Therefore, it was necessary to find a way to transform the LOGTAB table into some computer table that did not require much storage and which could be interpreted fast. The following format is the result:

A given table is expressed in the computer in three distinct but highly interrelated parts. Part I is a table of values of the different elements of a row of the original table. By entering only the different elements, it is possible to take advantage of redundancy in the original. Part II is a table of the same length as Part I with a one-to-one correspondence between their entries. It contains a bit pattern showing in which column or columns the value in part I appears. A particular bit position in a Part II entry corresponds to a column in the original table. Part III consists of a group of calling sequences (in 704 language) to subroutines which look for matches between the input data and the table entries and then extract the proper results and act on them as specified in the table.

The principle can best be shown by an example. Figure 4 is a table which is designed to show two of the machining operations to be performed on a bucket with certain characteristics (dovetail type, tenon shape (rectangular or profile), and nature of the dovetail sides.) Expressed as a LOGTAB table it takes the form shown in Figure 5. Note that some, but not all, of the redundancy in the original has been reduced in changing it to LOGTAB form but that it is still humanly readable. Reading it in English it says: If dovetail sides (DS) = ANGLE and if TENON = RECT and if the dovetail type (DT) = 22 or 23 or 24 or 25 or 43 or 48 or 49, then operation No. 2 (OP2) = 25. If at least one of these conditions was not met, then we go on to the next column, etc. Quadrant D **'s have been omitted.

Note that in Figure 4 there are only eight possible solutions and these are indicated by asterisks. These eight solutions are now shown as the eight columns in Figure 5.

Figure 6 shows Part I and Part II of the computer representation of the LOGTAB table. Looking at the table for dovetail side (DS) we see that it contains only the two different entries that can appear as compared to 80 entries in the original and 8 in the LOGTAB form. Their distribution within the LOGTAB table is shown by the corresponding bit pattern in Part II. ANGLE which is the entry in columns 1, 2, 5 and 6 is represented by 1's in bit positions 1, 2, 5, and 6.

To read or interpret the table we must find the intersection or logical product of the expressions in Part II corresponding to matches made between input data and elements in Part I. This logical product is then matched with the Part II representation of the result table to find the correct solution.

As an example, take the following input values: DS = ANGLE, TENON = RECT, and DT = 72. The bit pattern for DS = ANGLE is 11001100 and for TENON = RECT it is 10101010. The logical product of these two expressions is 10001000 and therefore our final result must be in column 1 or column 5. Next we find that the bit pattern for DT = 72 is 00001111 and the logical product of it and our previous product is 00001000. This means that column 5 of our LOGTAB table contains the solution we are seeking. Continuing with our bit pattern, we match it to the entries for OP1 again by a logical product.

```
00001000  =  input
00001111  =  No. 23
--------
00001000  =  product
```

The product for (OP1 = 23) is not zero and, therefore, OP1 = 23 is the desired result. In forming the same product with entries for OP2 we find that the desired result is OP2 = 25.

It is possible that with some tables the input will yield a multiple solution bit pattern (i.e., more than one 1-bit in the logical product). But our LOGTAB tables are so constructed that when one column in the B quadrant matches our input we do not test the other columns. Therefore, to find a unique solution we only have to change all 1-bits after the first one in the product bit pattern to zeroes.

Part III for the example table consists of calling sequences to subroutines to do the following steps:

1 - Set bit pattern for result to all 1's.

2 - Get input value for DS and go to routine to find DS match in Part I and compute logical product of bit pattern result and corresponding DS bit pattern.

3 - Same as 2 for TENON

4 - Same as 2 for DT

5 - Go to routine to set bit pattern result for unique solution (preserve only first 1-bit).

6 - Go to routine to find OP1 by first non-zero logical product of bit pattern for result and bit pattern for various OP1 values.

7 - Same as 6 for OP2.

The combined computer expressions in Parts I, II and III require only about 100 words compared to 136 entries in the LOGTAB table (quadrants B and D) and 400 entries in the original table. The complete interpretation was equivalent to executing approximately 200 704 instructions. This corresponds to executing about 1.5 program steps for each entry in the LOGTAB table (B and D quadrants only) and only 0.5 steps per entry in the original table. In actual time, it requires about 5 milliseconds to find a solution in this original table on the 704.

The subroutines to make all of the interpretations require a total of about 160 words of memory but they are used for every table and therefore appear only once. For speed there are separate subroutines for making the tests for the various conditions ($\angle$, $\leq$, etc.). It is for this purpose and for simplifying the interpretation that the LOGTAB tables may test for only one condition per row (i.e., all values in a row must be tested for the same condition).

A compiler is planned and is presently being programmed which will accept a LOGTAB table on punched cards and translate it into the computer representation of Parts I, II, and III.

DVT TYPE	DVT SIDES	TENON	OPER NO.1	OPER NO.2	DVT TYPE	DVT SIDES	TENON	OPER NO.1	OPER NO.2
* 22	ANGLE	RECT	-	25	49	ANGLE	RECT	-	25
* 22	ANGLE	PROF	-	21	49	ANGLE	PROF	-	21
* 22	NO	RECT	-	22	49	NO	RECT	-	22
* 22	NO	PROF	-	24	49	NO	PROF	-	24
23	ANGLE	RECT	-	25	50	ANGLE	RECT	23	25
23	ANGLE	PROF	-	21	50	ANGLE	PROF	23	21
23	NO	RECT	-	22	50	NO	RECT	23	22
23	NO	PROF	-	24	50	NO	PROF	23	24
24	ANGLE	RECT	-	25	53	ANGLE	RECT	23	25
24	ANGLE	PROF	-	21	53	ANGLE	PROF	23	21
24	NO	RECT	-	22	53	NO	RECT	23	22
24	NO	PROF	-	24	53	NO	PROF	23	24
25	ANGLE	RECT	-	25	62	ANGLE	RECT	23	25
25	ANGLE	PROF	-	21	62	ANGLE	PROF	23	21
25	NO	RECT	-	22	62	NO	RECT	23	22
25	NO	PROF	-	24	62	NO	PROF	23	24
* 42	ANGLE	RECT	23	25	64	ANGLE	RECT	23	25
* 42	ANGLE	PROF	23	21	64	ANGLE	PROF	23	21
* 42	NO	RECT	23	22	64	NO	RECT	23	22
* 42	NO	PROF	23	24	64	NO	PROF	23	24
43	ANGLE	RECT	-	25	65	ANGLE	RECT	23	25
43	ANGLE	PROF	-	21	65	ANGLE	PROF	23	21
43	NO	RECT	-	22	65	NO	RECT	23	22
43	NO	PROF	-	24	65	NO	PROF	23	24
44	ANGLE	RECT	23	25	70	ANGLE	RECT	23	25
44	ANGLE	PROF	23	21	70	ANGLE	PROF	23	21
44	NO	RECT	23	22	70	NO	RECT	23	22
44	NO	PROF	23	24	70	NO	PROF	23	24
46	ANGLE	RECT	23	25	71	ANGLE	RECT	23	25
46	ANGLE	PROF	23	21	71	ANGLE	PROF	23	21
46	NO	RECT	23	22	71	NO	RECT	23	22
46	NO	PROF	23	24	71	NO	PROF	23	24
47	ANGLE	RECT	23	25	72	ANGLE	RECT	23	25
47	ANGLE	PROF	23	21	72	ANGLE	PROF	23	21
47	NO	RECT	23	22	72	NO	RECT	23	22
47	NO	PROF	23	24	72	NO	PROF	23	24
48	ANGLE	RECT	-	25	74	ANGLE	RECT	23	25
48	ANGLE	PROF	-	21	74	ANGLE	PROF	23	21
48	NO	RECT	-	22	74	NO	RECT	23	22
48	NO	PROF	-	24	74	NO	PROF	23	24

FIGURE 4

DS	=	ANGLE	ANGLE	NO	NO	ANGLE	ANGLE	NO	NO
TENON	=	RECT	PROF	RECT	PROF	RECT	PROF	RECT	PROF
DT	=	22	22	22	22	42	42	42	42
		23	23	23	23	44	44	44	44
		24	24	24	24	46	46	46	46
		25	25	25	25	47	47	47	47
		43	43	43	43	50	50	50	50
		48	48	48	48	53	53	53	53
		49	49	49	49	62	62	62	62
		-	-	-	-	64	64	64	64
		-	-	-	-	65	65	65	65
		-	-	-	-	70	70	70	70
		-	-	-	+	71	71	71	71
		-	-	-	-	72	72	72	72
		-	-	-	-	74	74	74	74
OP1		-	-	-	-	23	23	23	23
OP2		25	21	22	24	25	21	22	24

FIGURE 5

PART I		PART II
DS	ANGLE	1 1 0 0 1 1 0 0
	NO	0 0 1 1 0 0 1 1
TENON	RECT	1 0 1 0 1 0 1 0
	PROF	0 1 0 1 0 1 0 1
DT	22	1 1 1 1 0 0 0 0
	23	1 1 1 1 0 0 0 0
	24	1 1 1 1 0 0 0 0
	25	1 1 1 1 0 0 0 0
	42	0 0 0 0 1 1 1 1
	43	1 1 1 1 0 0 0 0
	44	0 0 0 0 1 1 1 1
	46	0 0 0 0 1 1 1 1
	47	0 0 0 0 1 1 1 1
	48	1 1 1 1 0 0 0 0
	49	1 1 1 1 0 0 0 0
	50	0 0 0 0 1 1 1 1
	53	0 0 0 0 1 1 1 1
	62	0 0 0 0 1 1 1 1
	64	0 0 0 0 1 1 1 1
	65	0 0 0 0 1 1 1 1
	70	0 0 0 0 1 1 1 1
	71	0 0 0 0 1 1 1 1
	72	0 0 0 0 1 1 1 1
	74	0 0 0 0 1 1 1 1
OP1	23	0 0 0 0 1 1 1 1
	-	1 1 1 1 0 0 0 0
OP2	21	0 1 0 0 0 1 0 0
	22	0 0 1 0 0 0 1 0
	24	0 0 0 1 0 0 0 1
	25	1 0 0 0 1 0 0 0

FIGURE 6