INTEGRATED
SYSTEMS
PROJECT

REPORT

TABSOL

A GENERALIZED SOLUTION TO PROGRAMING DECISION SYSTEMS

MARCH 21, 1959

R59MS301

MANUFACTURING SERVICES

GENERAL ELECTRIC

NEW YORK, N. Y.

GENERAL ELECTRIC

TECHNICAL INFORMATION SERIES

Title Page

AUTHOR B Grad TF Kavanagh	SUBJECT CLASSIFICATION Computers, Automatic Programming	NO. R 59 MS 301 DATE March 21,1959

TITLE

TABSOL - a generalized solution to programming decision systems

ABSTRACT TABSOL is a unique concept for formulating and solving decision systems. Through the use of a special form called a decision structure table the problem logic can be readily expressed by a systems designer without specific computer programming knowledge. This generalized format can then be solved automatically by an interpretive computer program.

G.E. CLASS 3 GOV. CLASS.	REPRODUCIBLE COPY FILED AT Production Control Service New York, New York	NO. PAGES 31

CONCLUSIONS

TABSOL appears to be a major step forward in automatic computer programming since it replaces both the coding and the flow charting. In addition, the format chosen tends to highlight logical errors. The general concept of TABSOL looks very promising in other areas: scientific calculations, specialized function programming, etc. This material should not be reproduced without specific permission of Production Control Service. The material is also Company Confidential and should not be referred to with any non-General Electric employees.

INFORMATION PREPARED FOR	
TESTS MADE BY	Production Control Service
AUTHOR	B. Grad, T. F. Kavanagh
COUNTERSIGNED	H. F. Dickie, Mgr. - Production Control Service
SECTION	Materials Service
LOCATION	New York, New York

Table of Contents

(1) The concept of a logical table

TABSOL is a sub-routine for "solving" logical tables. A logical table is a coded representation of a decision-making system. It lists a set of argument variables and several sets of result functions; only one set of result functions, for example, is to be executed, the choice of result set being dependent upon the current values of the argument variables. In the type of table that TABSOL is able to solve, to execute a set of functions means simply to assign values, called results, to other variables, called function variables.

A logical table, therefore, consists of a list of argument variables, several sets of tests, several sets of results (one set for each set of tests) and a list of function variables. Figure 1(a) below shows how these entries are arranged to form a table. Figure 1(b) shows a sample table, in an uncoded form; in actual practice each entry in the table would be expressed in five digits of a special TABSOL code.

The first column contains the argument variables (above the double line) and the function variables (below the double line). The second column, and each column thereafter, contains a set of tests (above the double line) and a set of results (below the double line). To solve this table, TABSOL would ascertain the current values of each argument variable, and then apply to each argument values the test indicated in the second column. If all the tests are satisfied, the second column is the solution column; if not, the tests in the succeeding columns are applied, until the solution is found. TABSOL then takes the results in the lower part of the colution column, and assigns each result as the value of the function variable indicated in column 1. Note that, as shown in the bottom row of the example, it is possible to make the number of the table to be solved next depend upon the solution found for the last table solved. Thus, one can solve table after table without leaving the TABSOL sub-routine.

col 1	col 2	col 3		col n
Argument variables	1st set of tests	2nd set of tests		(n-1)'th set of tests
Function variables	1st set of results	2nd set of results		(n-1)'th set of results

Figure 1(a)

Quantity, rating, diameter, and tolerance of capacitor for AC wattmeter.

Voltage rating	⩾ 100	⩾ 100	⩾ 100	⩾ 200	⩾ 200	⩾ 200
Voltage rating	⩽ 125	⩽ 125	⩽ 125	⩽ 250	⩽ 250	⩽ 250
Number of phases	= 1	= 3	= 3	= 1	= 3	= 3
Shock characteristic	-	Hi shock	Comm.	-	Hi shock	Comm.
Quantity	1	2	2	1	2	2
Rating in μF	.039	.027	.039	.027	.022	.027
Diameter, in inches	7/16	7/16	7/16	3/8	7/16	7/16
Tolerance (%)	5	10	5	10	-	10
# table to solve next	004	017	017	004	017	014

Figure 1(b)

(2) How the variable data fields are set up for use by TABSOL.

(i) The variable data upon which TABSOL operates is broken up into fields of fixed length. A field may be numeric (i.e. each digit is an unsigned number 0 through 9) or alphameric. An alphameric field is defined as any field that is not numeric. It may contain any number, letter of the alphabet, or other typewriter character, except that the first character may not be &, ., ⌑ , -, or $.

(ii) The variable data fields are divided into 3 "lists", as shown below. There is also a print image area, where results may be stored, but from which arguments cannot be taken.

List name	Symbol	Field length and type	Contents
SPEC	-	4-digit numeric or alphameric	Customer specifications Product characteristics, etc.
INT	&	4-digit numeric or alphameric	Intermediate characteristics
TIM	$	5-digit numeric	Time values
PRNT	P	4-digit alphameric or numeric	Results required for printing only, not for future reference.
	Q	5-digit numeric	

(iii) To address a field in the Spec, Int, or Time lists, one gives the list symbol followed by the serial entry number of the field within the given list: -0005, &0127, $0063. In a few instances only 4 characters are available in which to express the address; in these cases, the leading zero of the entry number is dropped, and &0127 is written &127. It follows that the maximum permissible entry number is 0999. The lowest entry number is 0001, not 0000.

(iv) The print image area has no fixed format; a four- or five-digit result may be stored anywhere in the image. One addresses the area by giving the symbol P (or Q) followed by the number of the print image position in which the right-most digit of the result is to be stored: P0025, Q0167, P1204.

The maximum permissible image position number is 9999. Code P, the one most commonly used, causes a 4-digit result to be stored; code Q causes a 5-digit result to be stored, and should be used only with numeric results. The executive program must supply the record mark needed to define the division of the print area into lines.

(v) The location within memory of the lists and the print area is not fixed by TABSOL. The executive program informs TABSOL of the starting location assigned for each area.

(3) The TABSOL coding system

We list and explain below the permissible codes that may appear in the argument, test, result and function areas of a table. Each table entry is five digits in length.

(a) Arguments

May be the name of a field in the SPEC, INT, or TIM areas only, e.g. &0127, the 127th field in the INT list. If the argument is a TIM field,, the tests appearing in the same row must be comparisons to other TIM fields; errors will result from any attempt to compare a five-digit TIM field to a four-digit SPEC, INT or literal field.

(b) Tests

(i) The first digit of a test entry is the comparison type-code, showing what type of comparison is to be made between the argument variable and the test value. The possible type-codes are:

Symbol	Comparison Indicated	
b	$\equiv$	identical equivalent
G	$>$	greater than
H	$\geqslant$	greater than or equal to
L	$<$	less than
M	$\leqslant$	less than or equal to
U	Make no comparison, skip to do next test.	
Z	Make no more comparisons, this column is the solution column.	

(ii) The last four digits of a test entry either give the literal test value, or give the address of a variable to be used as a test value. The latter case is called a non-literal value. Thus, the last four digits may be:

1234	-	numeric literal
VANE	-	alphameric literal
-005 &127 $063	-	non-literals

(iii) The argument is always compared to the test value, i.e. H0200 means "is the argument $\geqslant$ 0200".

(iv) Note that a time field may be used as a non-literal test value only if the corresponding argument is also a time-field.

(v) Note also that the comparison type-code Z may occur only in the top row of the last column. It indicates an "all-other" column.

(vi) Since there are only four characters available in which to write a non-literal test value, the leading zero of the entry number is dropped, i.e. &0127 becomes &127.

(c) Results

(i) A result, like a test value, may be literal or non-literal. Literal results occurring opposite a TIM function must be five-digit numerics. Literal results occurring opposite a PRNT function may be a four-digit numeric or alphameric (e.g. b1234, bVANE), or five digit numeric (e.g. 12345, 00654). We list the possible types of result fields below:

b 1234	4-digit numeric literal
b VANE	4-digit alphameric literal

12345	5-digit numeric literal; can occur only opposite TIM or Q functions.
&0127	4-digit non-literal
-0005	4-digit non-literal
$0063	5 digit non-literal, can occur only opposite TIM or Q functions.
U bbbb	no result to be stored, skip to do next result row.
Z bbbb	no more results, skip to solve next table, as shown in director.
F 0012	the result value is to be found by evaluating a formula (see below for further explanation).

(ii) When a result is not equal to a known literal or variable, but must be found by evaluating a formula, the necessary "pseudo-instructions" are written immediately following the table. The result entry in the table shows the number of the pseudo-instruction at which to start, i.e. F0012 means transfer to execute pseudo-instruction 12 and continue executing pseudo-instructions until a "last instruction" code is encountered. Further details are given in the section on Pseudo-Arithmetic.

(d) Functions

(i) A function may be any of the following:

-0005 &0127 $0063	address of function variable to which result value is to be assigned.
P0025 Q0025	store result value in print image area, with right-most digit of result going into 25th position of print image.
#bbbb	the result value is the number of the table to be solved next. In this case the result must be literal and be $\leqslant$ 999.

(ii) Whenever a result is stored as a time variable, it is also added into a six-digit "total time accumulator". The executive program is responsible for re-setting this accumulator to zero whenever necessary. Any overflow beyond six digits will be lost.

(iii) A TIM non-literal result must not occur opposite a SPEC or INT function. However, a SPEC or INT non-literal result may appear opposite a TIM function; a leading zero will be added to the result value before it is stored.

(iv) Similarly, a SPEC or INT non-literal result may appear opposite a PRNT function; a leading zero will be added, and the result will be treated as five-digit.

(v) The method of storing the result in the print area depends upon whether the function code is P or Q.

Code P causes the right-most four digits of the result value to be placed in the print area without change.

Code Q causes all five digits of the result value to be placed in the print area by means of a "store-for-print" instruction. Storing starts with the right-most digit of the result; any decimal points or commas encountered in the print image are skipped over, and any zeros or commas to the left of the most significant figure are obliterated and replaced by blanks. Zeros occurring to the right of a decimal point are not obliterated. Also the position to the right of the one addressed by the print function is automatically set to a blank; any character previously placed there will be lost.

We show some examples on the following page.

Function	Result Value	Print image before	Print image after
		$19 \longrightarrow 26$	$19 \longrightarrow 26$
P 0025	12345	bbbbbbbb	bbb2345b
Q 0025	12345	bbbbbbbb	bb12345b
Q 0025	12345	b, bbb. bb	1, 234. 5b
P 0025	00789	bbbbbbbA	bbb0789A
Q 0025	00789	bbbbb. bA	bbb78. 9b

The print functions may have any value from P 0004 to P 9999, or from Q 0005 to Q 9999.

The "store-for-print" instruction works correctly only if the quantity to be stored is numeric. Using print code Q in conjunction with an alphameric result will cause unpredictable errors in the field actually stored in the print image

(vi) When it is desired to return control to the executive routine, the result field bbbb* is written opposite the # function. The # code may occur <u>only</u> in the last row of a table.

(4) The TABSOL coding system: table descriptors

(i) Each table is immediately preceded by a 14-digit "descriptor", containing the following information about the table:

GGNNN	-	identification number of table. GG is the table group number, NNN number of the table within the group.
RR	-	total number of rows (argument and function).
CC	-	number of columns, including the first.
AA	-	number of argument rows, i.e. number of rows above the double line.
DDD	-	the "director".

The fields are written in the order given above.

(ii) The director is normally used only when the last function of the table is not #. In this case, the director contains the three-digit number of the next table to be solved. If it contains bb*, control will be returned instead to the executive, as described above.

For a table having a # function, the director shows the number of the table to be solved next if it is found that the table currently being solved has no solution, i.e. none of the sets of tests is satisfied.

(5) The TABSOL coding system: pseudo-arithmetic

(i) All pseudo instructions are five characters in length. The first character is an operation code, describing the type of operation to be performed. The remaining 4 characters are the operand, i.e. the field upon which the operation is to be performed. The operand may be literal, e.g. 1234 or non-literal. Non-literal operands may refer to the Spec, Int, or Tim lists, or to a "holding" area. There are five holding areas, H001 through H005; they are used for temporary storage of intermediate sums or products.

(ii) The actual operand, by which we mean either a literal operand or the field stored at the address given in a non-literal operand, must be numeric. A non-numeric operand causes an immediate exit from TABSOL to the error return location in the executive program.

Actual operands may be either 4-digit (literals, or fields from the Specs or Int lists) or 5-digit (fields from the Tim list or from a holding area). Four-digit operands are treated as the four least significant digits of a five-digit operand, i.e. a leading zero is added before the pseudo-operation is executed.

(iii) The results of pseudo-operations appear in a register called the pseudo-accumulator (PAC). PAC is ten digits long. Addition and subtraction are done in the right-most five digits only; if any six-digit sum is formed, the left-most digit of the sum will be lost. All operands are unsigned and assumed to be positive; if any sum of negative value is formed, the negative sign will be lost.

(iv) We list below the 11 pseudo operation-codes. In the following, Y sumbolizes the five-digit actual operand, as defined in (ii) above. PAC_x means the x^{th} digit of the PAC register.

Code	Name	Effect
B	Bring	Clear PAC_{1-10}. Place Y in PAC_{6-10}.
A	Add	Add Y to contents of PAC_{6-10}; store sum in PAC_{6-10}.
S	Subtract	Subtract Y from contents of PAC_{6-10}; store result in PAC_{6-10}.
M	Multiply	Multiply Y by contents of PAC_{6-10}; store ten-digit product in PAC_{1-10}.
D	Divide	Divide contents of PAC_{1-10} by Y; store ten-digit quotient in PAC_{1-10}.
L	Left shift	Shift contents of PAC_{1-10} to the left Y places. Digits shifted to the left of PAC_1 are lost. Replace shifted digits by 0.
R	Right shift and Round	Shift contents of PAC_{1-10} Y places to the right. Digits shifted to the right of PAC_{10} are used to round the digit in PAC_{10}, then dropped.
Q	sQuare root	Find square root of number in PAC_{1-10}, assuming a decimal point PAC_5 and PAC_6. Store result back in PAC_{1-10}, with decimal point similarly placed.
H	Hold	Store contents PAC_{6-10} in hold entry Y. Y must be 1, 2, 3, 4, or 5.
E	Transfer	Transfer to execute pseudo-instruction number Y.
X	eXit	The number now in PAC_{6-10} is the value of the formula. Return to the main routine to store this value as directed.

Operation codes Q and X must have an operand of 0000.

Operation codes H and E must have literal operands.

(v) We show on the following page the changing contents of PAC as a series of pseudo-instructions is executed.

Instruction	Actual operand	PAC	Comments
B 9876	09876	00000 09876	
A $024	54321	00000 64197	
A $007	44444	00000 08641	Six-digit sum; high-order digit lost.
S 1234	01234	00000 07407	
S -063	09639	00000 02232	Negative result; sign lost.
L 0004	-	00223 20000	
D &127	00192	00001 16250	
R 0002	-	00000 01163	
H 0003	-	00000 01163	01163 stored at 3rd hold entry.
B $009	00225	00000 00225	
L &077	00005	00225 00000	Shift may have non-literal operand.
Q 0000	-	00015 00000	
R &077	00005	00000 00015	
A H003	01163	00000 01177	
B $007	44444	00000 44444	
D $019	23432	00000 00002	Note need to shift dividend left before dividing to obtain sufficient number of digits in quotient.

During the execution of a divide or square root instruction the answer is computed to one more place than is shown in the final result stored in PAC. This result is obtained by rounding off the extra digit.

(vi) The sequence of four instructions shown for finding the square root of 225 could also be coded thus:

B $007	00000 00225	
L 0001	00000 02250	$= 225 \times 10^{-4}$
Q 0000	00000 15000	$= \sqrt{225} \times 10^{-2}$
R 0003	00000 00015	

Provided that the actual decimal point is an even number of places away from the assumed decimal point, the correct answer will result, though it will be shifted from its true position in PAC.

(vii) If a zero divisor is detected when a divide pseudo-instruction is about to be executed, the divide operation will not be performed, and control will be returned to the error return location in the executive.

(viii) The contents of PAC and the hold areas are never cleared out by TABSOL, i.e. they are unchanged except by a pseudo-instruction that operates on their contents.

(ix) When a formula evaluation is completed, a five-digit result is obtained. If the function corresponding to this result is a SPEC or INT or P function, only the right-most four digits of the result value will be stored. If the function is a Time or Q function, all five digits are stored.

(6) Information to be supplied to TABSOL by the executive program

(i) At the end of TABSOL space has been left for various pieces of information that must be supplied by the executive before TABSOL is entered. We show below their names, symbolic locations, actual locations (relative to F, the starting location of TABSOL) and significance:

Name	Symbolic Loc.	Significance
TAB	59.90.0	Location of 1st digit of 1st table in group. (automatically set to 10,001 unless specified otherwise)
SPEC	59.91.0	(Location of 1st digit of spec. list) -1
INT	59.92.0	(Location of 1st digit of int. list) -1
TIM	59.93.0	(Location of 1st digit of time list) -1
PRNT	59.94.0	(Location of 1st digit of print image) -1
RET	59.95 0	Re-entry location in executive program.
ID	59.96.0	5-digit identification number of table to be solved.

(ii) The first five quantities will normally be fixed for any given executive program. If TABSOL is to be assembled along with the executive, it is suggested that the actual values assigned be punched in the symbolic TABSOL cards.

The last quantity, ID, will normally be reset each time TABSOL is entered from the executive.

The return address, RET, may be fixed (and punched) before assembly, or reset at each entry to TABSOL.

(iii) If TABSOL recognizes an error condition (see below) during the solution of a table, it will return control to the instruction at RET + 5 in the executive.

(iv) The six addresses mentioned may be in either lower memory or upper memory, as desired. However, the tables themselves must lie either wholly in lower memory or wholly in upper memory. Hence, a group of tables may not exceed 9999 digits in size.

(v) TABSOL has no ability to pass from tables in one group to tables in another group. To accomplish this, control must be returned to the executive (by means of the asterisk code), which reads in the new group of tables and the new dictionary for the group, resets all five digits of ID, and then re-enters TABSOL.

(vi) It is assumed that the dictionary follows TABSOL. The dictionary must be in lower memory.

(7) Dictionary of Table Location Increments

(i) The executive is responsible for bringing into memory the tables themselves, and also a "dictionary" showing the starting location of each table relative to TAB, the starting location of the first table.

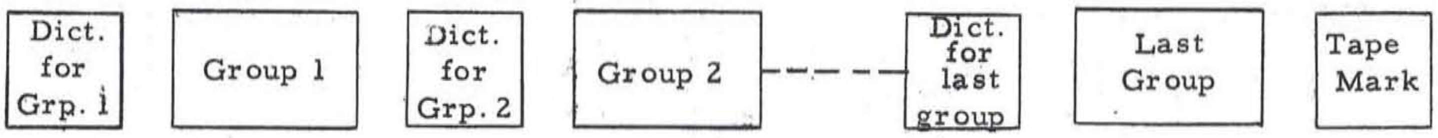
A dictionary consists of a series of 4-digit entries, one for each table in the given group. If a particular table is missing, i. e. table 3 is followed directly by table 5, a dictionary entry of 0000 must be written for the missing table. The dictionary entry for a given table is computed by summing the sizes of the preceding tables in the group.

Example of dictionary:	0000	-	entry for table 1.
	0127	-	entry for table 2
	0365	-	entry for table 3
	0000	-	entry for missing table 4
	0483	-	entry for table 5
	----	-	---------------
	9876	-	entry for last table in group.

(ii) Since a group of tables cannot exceed 9999 characters, the largest table location increment possible is somewhat less than 9999.

(iii) The dictionary normally follows immediately after TABSOL in memory. If this is inconvenient, insert the desired symbolic (or actual) starting location minus 1 in the address (or value) field of card 59. 09. 0. Do not omit the plus sign in column 19 of this card.

(8) Layout of Table Tape



(i) Each group of tables (a group being any convenient bunch of less than 10,000 digits in size) forms one record on the table tape. Each group is preceded on the tape by the dictionary for that group; the dictionary also forms one record.

(ii) A tape mark exists on the tape after the last group of tables, i.e. there is only one file of information on the tape.

(iii) It takes approximately .72 seconds to read one dictionary and one group of tables into memory, assuming that the group contains 100 tables and is 9999 digits long.

(iv) TABSOL is unable to pass from a table in one group to a table in another group. It is the responsibility of the executive program to read a new group of tables into memory.

(v) The first five digits of any group of tables are the identification number of the first table in the group, e.g. 07001. This field, or the first two digits thereof, can therefore be used to identify a group.

(9) Size and timing of TABSOL

(i) TABSOL occupies 2332 locations in memory. This includes instructions, working storage, program constants and the basic information supplied to TABSOL by the executive. It does not include the space occupied by the dictionary, which is variable.

(ii) TABSOL takes about 10 milli-seconds to do each of the following:

(a) Prepare to solve the next table.

(b) Make one test.

(c) Store one result.

(d) Do one pseudo-instruction.

If the test or result code is U (make no test, store no result) the time required is about 5 milli-seconds.

(iii)

	A	B	B	B
	U	C	D	D
	U	U	E	F
	F	F	F	F

Change in test is shown by change in letter.
U indicates "make no test".

The average time taken to solve the above table, supposing the solution to lie with equal frequency in each column, is computer as follows. (We suppose that one of the results is always a six-instruction formula.)

Set-up time:	10 m. s.
Average test time:	1/4 (2 + 3. 5 + 6 + 9) x 10 $\simeq$ 50 m. s.

Result time:		40 m. s.
Pseudo-op time:		60 m. s.
∴ Total time:	=	160 m. s.

(10) Errors recognized by TABSOL

(i) TABSOL recognizes 4 different types of error. Each one of them causes a 10-digit error message to be placed in the "error cause" location with TABSOL, and then returns control to the "error re-entry" location in the executive. The identification number of the table currently being solved is also available to the executive at ID.

(ii) When the executive has taken appropriate action to record the occurrence of the error, it may wish to return to TABSOL and continue solving more tables in the same group. This is accomplished by re-entering TABSOL at symbolic location 57.01.0. The number of the next table to be solved will under these circumstances always be taken from the director, regardless of the presence or absence of the # function in the table.

(iii) The four possible types of error, and the resulting error messages are shown below:

(a) WTI - wrong table identification. The table number in the descriptor of the table about to be solved does not match the table number in ID, which is the number of the table it is desired to solve. This error is not probable, but could occur if the dictionary were incorrect or if TABSOL were directed to solve a table that is missing. The error message is:

WTIbb GGNNN

where GGNNN is the identification number found in the table descriptor.

(b) No solution. TABSOL has done all the columns of tests without finding a solution column. The error message is:

NO b SOLN bbb

(c) NNO - non-numeric operand. The actual operand of a pseudo-instruction is not a numeric field. The pseudo-instruction is not executed and the error message is:

F XXXX bb NNO

where F XXXX is the result entry currently being evaluated by pseudo-arithmetic. (e.g. F0001, F 0012)

(d) ZD - zero divisor. The actual operand of a pseudo divide instruction is zero. The division is not executed, and the error message is:

F XXXX bbb ZD

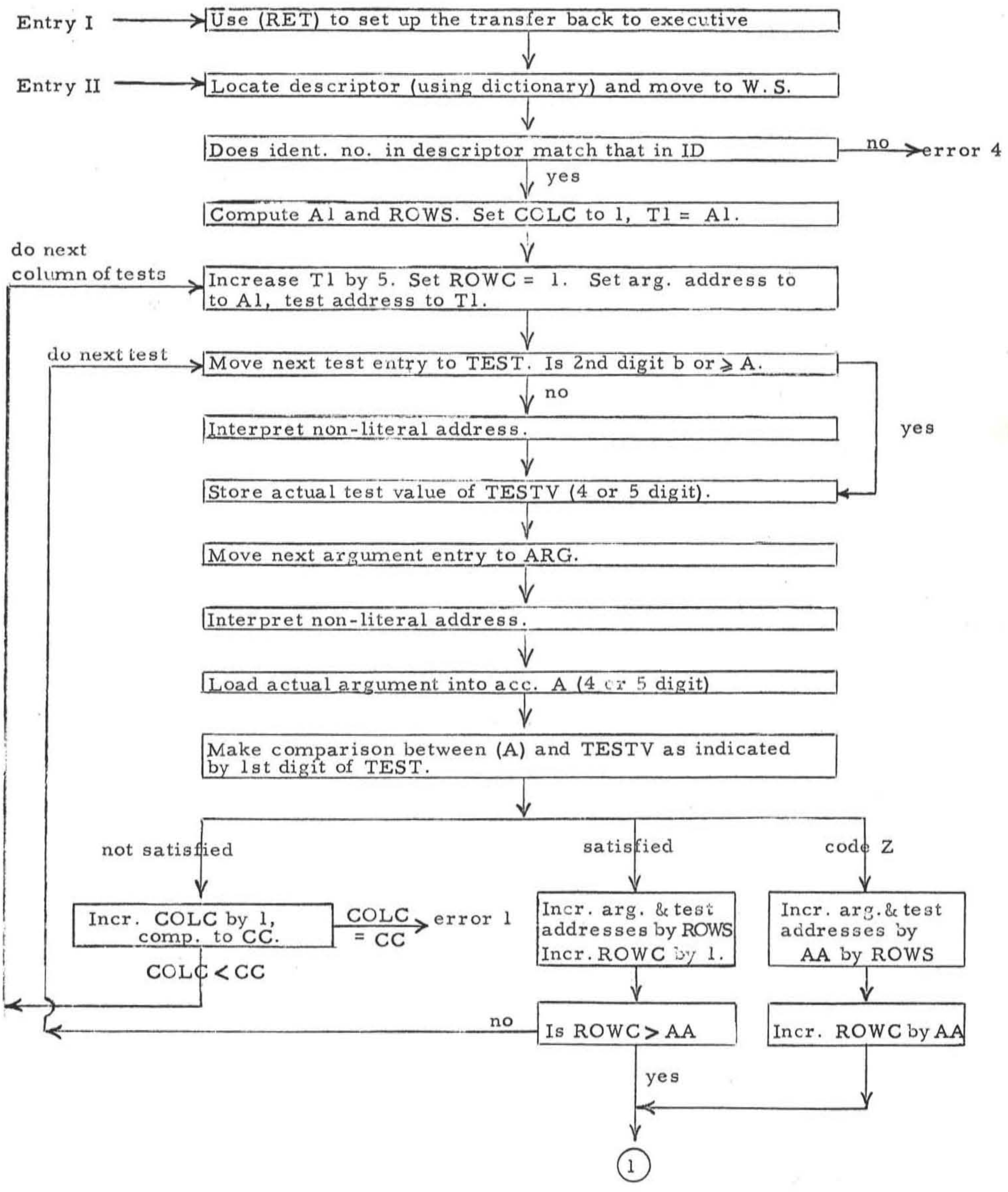
where F XXXX is again the result entry currently being evaluated.

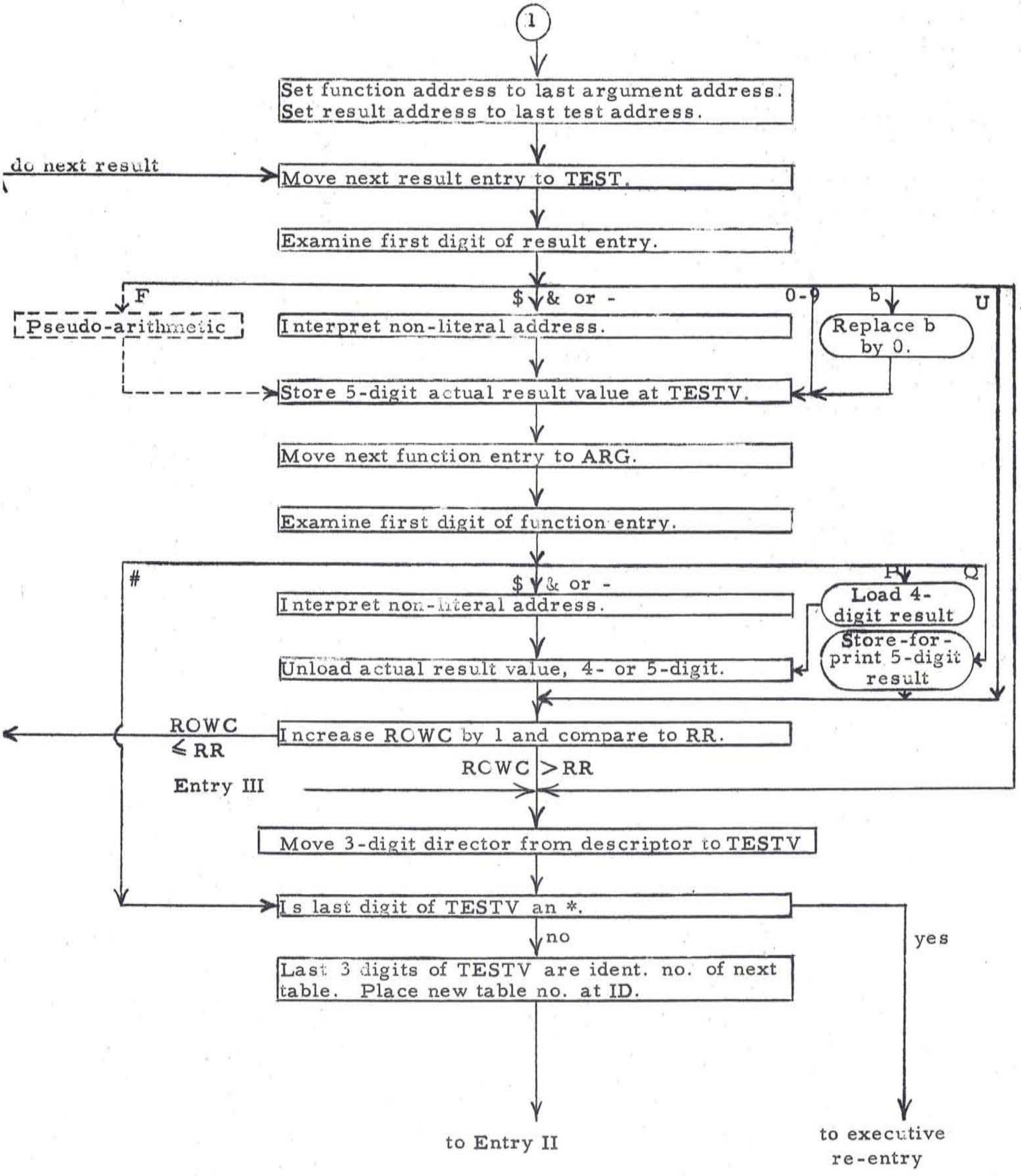
(11) Ready reference of TABSOL locations of interest to the executive

We list below all locations in TABSOL to which the executive program may have occasion to refer. The "card number" is a 3-digit field punched in columns 77-79 of each card in the symbolic TABSOL deck. All symbols refer to the right-most digit of a field.

Card No.	Symbolic Loc.	Length	Significance
001	50.0~~0~~ ~~9~~	5	Assigns the starting location for TABSOL assembly.
004	50.01.0	5	Entry point to TABSOL if RET changed since last entry, or if this is first entry made.
007	51.01.0	5	Entry point to TABSOL is RET unchanged since last entry.
401	57.01.0	5	Entry point after error has occurred; causes table whose number is given in director of current table to be solved next.
456	59.09.0	4	Address of 1st digit of dictionary -1.
479	59.61.0	14	Descriptor of table currently being solved.
487	59.69.0	10	Contains error message.
489	59.71.0	6	Total time accumulator (contains blanks originally).
490	59.90.0	4	TAB - address of 1st character of 1st table. Contains $00\bar{0}1$ unless punched otherwise.
491	59.91.0	4	SPEC - address of spec. area -1.
492	59.92.0	4	INT - address of int. area -1.
493	59.93.0	4	TIM - address of time area -1.
494	59.94.0	4	PRNT - address of print area -1.
495	59.95.0	4	RET - normal executive re-entry address.
496	59.96.0	5	ID - identification number of table currently being solved.

(12) Flow chart


Entry I
Use (RET) to set up the transfer back to executive
Entry II
Locate descriptor (using dictionary) and move to W. S.
Does ident. no. in descriptor match that in ID
no
error 4
yes
Compute A1 and ROWS. Set COLC to 1, T1 = A1.
do next column of tests
Increase T1 by 5. Set ROWC = 1. Set arg. address to to A1, test address to T1.
do next test
Move next test entry to TEST. Is 2nd digit b or ≥ A.
no
yes
Interpret non-literal address.
Store actual test value of TESTV (4 or 5 digit).
Move next argument entry to ARG.
Interpret non-literal address.
Load actual argument into acc. A (4 or 5 digit)
Make comparison between (A) and TESTV as indicated by 1st digit of TEST.
not satisfied
satisfied
code Z
Incr. COLC by 1, comp. to CC.
COLC = CC
error 1
COLC < CC
Incr. arg. & test addresses by ROWS Incr. ROWC by 1.
Incr. arg. & test addresses by AA by ROWS
no
Is ROWC > AA
Incr. ROWC by AA
yes
1


1
Set function address to last argument address. Set result address to last test address.
do next result
Move next result entry to TEST.
Examine first digit of result entry.
F
$ & or -
0-9
b
U
Z
Pseudo-arithmetic
Interpret non-literal address.
Replace b by 0.
Store 5-digit actual result value at TESTV.
Move next function entry to ARG.
Examine first digit of function entry.
#
$ & or -
P
Q
Interpret non-literal address.
Load 4-digit result
Store-for-print 5-digit result
Unload actual result value, 4- or 5-digit.
ROWC ≤ RR
Increase ROWC by 1 and compare to RR.
ROWC > RR
Entry III
Move 3-digit director from descriptor to TESTV
Is last digit of TESTV an *.
no
yes
Last 3 digits of TESTV are ident. no. of next table. Place new table no. at ID.
to Entry II
to executive re-entry

For an explanation of symbols used above see section (14), paragraph (v).

(13) Sample tables and data as used to check out TABSOL

(i) Information supplied to TABSOL

Loc.	Card Contents						
2647	3 8 $\overline{6}$ 1	3 5 $\overline{0}$ 0	3 5 $\overline{6}$ 0	3 6 $\overline{2}$ 0	3 7 $\overline{4}$ 0	3 4 $\overline{9}$ 4	0 4 0 0 1
	TAB	SPEC-1	INT-1	TIM-1	PRNT-1	RET	ID

(ii) Dictionary

Loc.	Card Contents		
2676	0 0 0 0	0 1 3 9	0 3 1 3

(iii) Variable data

Loc.	Card Contents					
35$\overline{0}$1	0101 | 0202 | COIL | 8000 | 0100 |					Spec. fields
35$\overline{6}$1	1234 | VANE | 5678 | 0000 | 0100 |					Int. fields
36$\overline{2}$1	23456 | 22222 | 33333 | 44444 | 55555 |					Time fields
37$\overline{4}$1	| 5 | 10 | 15 | 20 . | 25 | 30 , |					Skeleton print image
38$\overline{6}$0	‡					To define print image

Table 1

38$\overline{6}$1: 04001 | 05 | 05 | 03 | 003 $\longleftarrow$ causes table 3 to be solved next

36$\overline{7}$5:

-0002	U		H	0200	H	0202	Z	
&0001	L	1000	H	1240	M	1240		
-0001	U		U		M	0100		
-0006							b	4321
&0007							b	9999

Table 2

$40\overline{00}$: 04002 | 08 | 04 | 03 | 999 ← not used, since # function takes precedence

$40\overline{1}4$:

&0003		500	G	5000		5678
-0006	U		G	5000	L	5000
&0002	U		U			VANE
$0001					1	1111
P0012						BRAK
P0020					0	0148
&0007					U	
#						*

causes return to executive

Table 3

$41\overline{7}4$: 04003 | 07 | 05 | 02 | 002

$41\overline{8}6$:

&0007	L	-001	G	-001	G	-001	Z	
$0001	U			$005	L	$003		
&0006					-	0001		
P0032					$	0001		
-0007					F	0001		
$0002					F	0010		
#						002		

causes table 002 to be solved next

```
B-002 | M0025 | H0002 | B&007 | L0003 | D-001 | R0001 | SH002 | X0000

B$001 | M0007 | L0004 | Q0000 | R0002 | A&007 | X0000
```

(14) Snapshot

(i) Snapshot is the name of a useful technique for obtaining "pictures" of TABSOL's working storage area at certain preselected points during the execution of TABSOL. There are six of these points, and each one may be individually activated, i.e. caused to produce a snapshot. The method of activation is to insert one (sometimes two) cards at the back of the assembled TABSOL deck; the cards replace TABSOL instructions by transfers into the Snapshot routine.

Each snapshot includes working storage, executive input to TABSOL (i.e. TAB through ID) and the dictionary. Each snapshot is written on tape 8 as one record; the record will produce 2 or 3 lines of printed output, depending on the length of the dictionary.

After each snapshot has been taken, the program comes to a halt. The halt number is unique for each type of snapshot taken, and thus provides the programmer with a means of seeking, at the console, just what course TABSOL is taking during the solution of a particular table.

(ii) A symbolic SNAPSHOT deck is provided along with every TABSOL deck. It is punched on card nos. 497 - 535, occupies symbolic locations 59.96.1 - 59.99.9, and when assembled takes up 195 memory locations.

Card No. 497, symbolic loc. 59.96.1, assigns the starting location for the assembly of Snapshot. It is currently set to 9800, but can be changed if desired.

(iii) The "activator" cards must be prepared anew after each assembly, since they refer to locations within TABSOL and within Snapshot whose actual values are not known until after assembly.

We give below the instructions for preparing the activator cards. The real equivalent of the symbolic TABSOL address shown under "Locations", less 4, is to be punched in cols. 10-13 of the activator card. The real equivalent of the symbolic Snapshot address shown under "Address" is to be punched in cols. 17-20 of the activator card. The character shown under "Op." is to be punched in col. 16 of the activator card. The digits 05 should be punched in cols. 14 and 15 of <u>all</u> activator cards.

Loc.	Op.	Address	
51.27.0	1	59.96.2	Activator for halt 7003.
52.76.0	1	59.96.6	Activator for halt 7005.
52.68.0 52.91.0	K 1	59.97.0 59.96.9	Activator for halt 7006.
53.48.0	1	59.97.3	Activator for halt 7007.
57.24.0	1	59.97.8	Activator for halt 7008.
54.85.0	1	59.98.1	Activator for halt 7009.

(iv) The status of TABSOL at each halt is as follows:

7003	Set up done, i.e. table located in memory.
7005	One column of tests done, no solution found.
7006	One column of tests done, this is solution column.
7007	One result stored (does not occur if corresponding function is #).
7008	Table solved, ready to solve next table (does not occur if table results in a return to the executive).
7009	One pseudo-instruction done (does not occur after X or E instructions).

(v) Each snapshot shows the following:

Name	Length	Signed	Significance
HALT	4	Yes	Number of the halt that followed the snapshot.
COLC	2	Yes	Column counter.
ROWC	2	Yes	Row counter.
ROWS	4	Yes	Number of characters in one row of the table, i.e. 5 x CC.
OPER	5	Yes	Actual operand of pseudo-instruction, expanded to 5 digits and signed plus.
M	7	Yes	Number whose square root is to be found, reduced to 1st seven significant digits.
YI	7	Yes	Square root.
DESC	14	No	Descriptor of table currently being solved.
ARG	5	No	Argument or function field currently being interpreted.
TEST	5	No	Test or result field currently being interpreted.
INST	5	No	Pseudo-instruction currently being interpreted.
AI	4	No	Computer address of 1st argument in table.
TI	4	No	Computer address of top test field in column being done.
PAC	10	No	Psuedo-accumulator.
HOLD	25	No	The five "hold" locations.
ERRC	10	No	Most recent error message to occur.
TESV	5	No	The actual test value or result value of the test or result currently being done.
TTA	6	No	Total time accumulator.
TAB	4		As explained previously.
SPEC	4		As explained previously.

Name	Length	Signed	Significance
INT	4		As explained previously.
TIM	4		
PRNT	4		
RET	4		
ID	5		

Following ID comes the dictionary. It is the record mark at the end of the dictionary that defines the snapshot area; it is therefore essential that this record mark be present, even if the dictionary has been loaded from cards instead of read in from tape.

(vi) The snapshots are placed on tape 8. This number can easily be altered by changing the select instruction on card no. 522.

The halt after each snapshot can be made inoperative by changing the halt code on card no. 523 to a NO-OP code.

(vii) Assembling Snapshot along with TABSOL causes <u>no</u> alterations to TABSOL itself. The Snapshot cards are discarded, once Snapshot's usefulness has expired, and the remaining TABSOL deck is in perfect working order.

T. F. Kavanagh, Specialist
Production Control Service

Miss J. H. Kelly
Computer Usage Company, Inc.

8-21-58