

IBM Research Center
P. O. Box 218
Yorktown Heights, N. Y.

January 13, 1961

Subject: Tabular Techniques Development

In the past two years a number of people have explored the possibility of using tabular form as a means of expressing decision processes so as to present these logical decisions in a more understandable way. In order to keep IBM personnel acquainted with this area of development, we are planning to distribute appropriate material from time to time, reviewing current work in developing tabular techniques. The people receiving this material have been selected because of their interest in programming methods.

You may well ask, "What do you mean by tabular techniques?" The full meaning of these techniques will be described in the various papers to be distributed. For the present, let us define tabular techniques as being the use of a table form to present the decision logic or operating procedures. In other words, tabular techniques will present programming and system descriptions in a table format. The material we distribute will be of four types: (1) material obtained from customers experimenting with tabular form, (2) material obtained from the Committee on Data Systems Languages (CODASYL) concerning the work on tabular form development, (3) material produced within IBM describing technical developments or explaining the use of tabular form, and (4) material produced by competitors, describing their developments and applications.

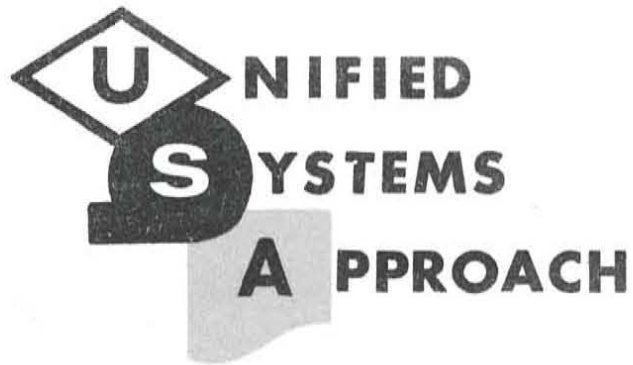
Since this is the first release, we would appreciate suggestions as to others who should be receiving this material, and any specific comments or ideas concerning the attached work. If you have any questions concerning these items, please call or write me.

This first distribution includes two items:

- (1) A status report on current developments in tabular techniques.
- (2) A copy of a speech given by Mr. T. F. Kavanagh, of General Electric, at the Eastern Joint Computer Conference on December 14, 1960. (Note that this speech differs from the paper printed in EJCC proceedings).



Burton Grad
Project Coordinator



CLEARINGHOUSE REPORT

TABULAR TECHNIQUES DEVELOPMENT
STATUS -- DECEMBER, 1960

January 13, 1961
Ref. No. 1A2

Burton Grad

INTERNATIONAL BUSINESS MACHINES CORPORATION
White Plains, New York

This material is distributed to keep IBM personnel informed of new developments. Selection is based on interest; this department makes no claim for the desirability of this approach nor necessarily recommends its use.

If additional copies are desired, please contact the Clearinghouse. No part of this material should be reproduced or distributed outside IBM without approval of the Clearinghouse.

December, 1960

Tabular Techniques Development Status

Burton Grad
IBM

During 1960 the use of tabular techniques in systems and programming languages has grown to become an area of significant experimentation. The one thing all these developments share is a tabular (or table) layout, in which the decision or program logic is recorded: information has positional significance as well as meaning contained within the statements.

However, just as machine languages are not the same, tabular techniques are not all the same. A number of people have contributed to these developments, and each person (or group) has followed a somewhat different path. Most of the developments have been limited to the particular application for which they were intended and have not been generalized.

This report serves to record in one place what has transpired during the past two or three years in the development of tabular techniques, and attempts to express major features and differences between various techniques.

Orren Evans, of Hunt Foods and Industries, first published work on using tabular form for computer programming. He had gained experience in the use of tabular form in his work with Sutherland and Company. The Hunt Foods material was released in December of 1959 to CODASYL, and later was presented at a Guide meeting and to the NMAA; it has been published by IBM as a General Information Manual. The decision structure tables are of a "limited entry" variety; this means that a complete condition or action statement is made in the stub (argument) of the table while the tabular entries only make assertions concerning the truth or falsity of the condition or indicate whether an action should be executed.

The Evans work corresponds in many ways to the Sutherland material. Copies of a current Sutherland proposal will be made available. Sutherland has prepared a number of tables describing a particular customer's decision rules; a 7070 program is being written from these tables instead of from flow charts. The Sutherland tables are still of the limited entry variety, though they are somewhat simpler in structure than the Hunt Foods' tables.

The CODASYL Systems Committee, prompted by the Hunt Foods work, has also decided to exploit tabular form to provide a systems-oriented language. I am a member of this Committee which also includes Les Calkins of U. S. Steel, Jack Strong of North American Aviation, Carl Byham of Southern Railway, Sol Pollack of RAND and some 8 - 10 others including representatives of RCA, Remington-Rand, and GE. The work which has been published to date in various intra-committee reports describes tabular form, data description, and certain systems-level operators. The tabular form material incorporates the limited entry approach of Hunt Foods, but also takes care of "extended entry" tables like those developed at General Electric: the table entries contain actual values, names or functions.

The General Electric work was initiated by the Integrated Systems Project in their Manufacturing Services Division (a staff group). As part of the major project aimed at designing an automatic factory, the need for describing complex, sequential, decision rules led this group to the creation and use of decision structure tables.

For variables (named fields) which have many values (more than two), the extended entry approach offers certain advantages; it is still quite easy to teach and relatively easy to implement. In contrast, the limited entry table may have substantial advantages for problems involving primarily two state variables. Up to recently, General Electric had not been willing to release any of the material which they have developed. However, at the last CODASYL meeting (12/1/60) Charlie Katz and Don Klick of General Electric's Computer Department, presented a paper, "Preliminary Reference Manual, TABSOL - 225 - A Tabular Systems Oriented Language for the GE 225 Information Processing System". This paper proposes a complete and quite comprehensive tabular form language which is to be directly processed on a GE 225. I should like to quote briefly from the introduction to this manual:

"Recent investigations by The Integrated Systems Project of General Electric's Manufacturing Services uncovered an area of applications which require neither extensive data file processing nor profound mathematics, but rather an unwieldy number of sequential decisions. To cope effectively with these decisions, the ISP team devised a tabular language. The purpose of this language was to depict, by means of tables, the relationships of logical decisions... Since its creation, TABSOL has been used in many departments of G. E. to analyze and solve problems of product engineering, manufacturing methods, cost accounting, and production control. The application of decision tables is continually growing. Recent studies show that they provide a concise method for supporting the logic of other data

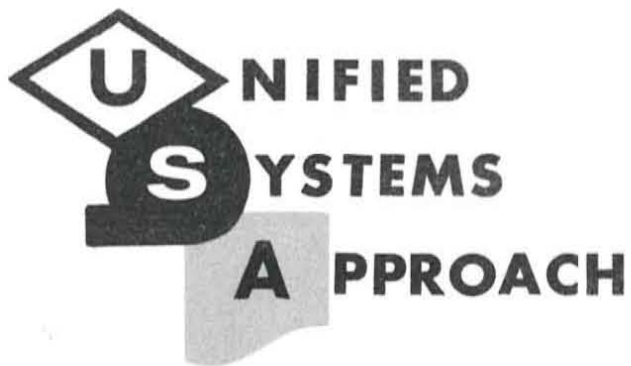
processing applications. For example, decision tables may be used to specify the transfer of control associated with the values of one or more fields, to control the printing of detail and summary lines of a report, or to interrogate the sort keys in a multi-file system. At the Computer Department we have found decision tables a valuable tool in designing and implementing the General Compiler."

"Decision tables represent a third language for the General Compiler. These may be used by themselves or in conjunction with the features of the compiler language. The specifications outlined in this manual pertain mainly to the table entries and imply and require a knowledge of the General Compiler ..."

General Electric has also permitted Mr. T. F. Kavanagh, who worked on the Integrated Systems Project, to present a paper entitled, "The TABSOL Concept" at the Eastern Joint Computer Conference on December 14, 1960. It is known that General Electric has probably thirty different departments (out of a total of 100) actually involved in experimenting on the practical use of decision tables. The specific experience of the ISP team is such as to indicate that the use of tables could save significant time in the programming and debugging of decision rules. Work in General Electric up through 1959 involved the preparation and use of interpreters for the 702, 704, 650, and 305 RAMAC. It would be reasonable to assume that in 1960, work on the NCR 304 would have progressed far enough to have a processor available, and there may be processors for other machines such as the Burroughs 205.

As a result of the CODASYL work, IBM was requested by North American Aviation to support their development work on tabular form. P. W. Knaplund, then Manager, Systems Marketing, DP Division, obtained the half-time services of M. D. Rayner who was assigned by R. V. Woodworth, then of the Inglewood office. Mr. Rayner is spending the other half of his time working with Northrop (Norair Division) on the development of another form of tables involving variable operation sequence. Neither of these programs are far enough along yet to have formal reports available.

W. M. Selden of IBM Corporate Systems Standards has been located at Rochester to work with Eastman Kodak in testing and developing concepts in the use of tables. Specifically, there are three projects either under way or ready to start there, with Eastman Kodak providing the bulk of the experimental work. One project has to do with the presentation of production control rules in their camera division. The second project has to do with the validation and updating of files in the Data Processing group. The third project is concerned with quality control decisions.



CLEARINGHOUSE REPORT

T A B S O L

A FUNDAMENTAL CONCEPT FOR
SYSTEMS ORIENTED LANGUAGES

Text of Speech Presented at E J C C
December 14, 1960

January 13, 1961
Ref. No. 1A1

T. F. Kavanagh
Manufacturing Services
General Electric Company

INTERNATIONAL BUSINESS MACHINES CORPORATION

White Plains, New York

This material is distributed to keep IBM personnel informed of new developments. Selection is based on interest; this department makes no claim for the desirability of this approach nor necessarily recommends its use.

If additional copies are desired, please contact the Clearinghouse. No part of this material should be reproduced or distributed outside IBM without approval of the Clearinghouse.

TABSOL

A Fundamental Concept for Systems Oriented Languages

T. F. Kavanagh
Manufacturing Services
General Electric Company

Bulging file cabinets, the flow chart jungle, mounting clerical costs, and the vast world where electronic computers haven't been successfully applied -- that's really what TABSOL and decision structure tables are all about. Structure tables have special meaning for information systems designers and programmers and they also have implications for hardware engineers because both computer user and computer designer must work together on the same information processing problems.

To date, the difficulties of communicating with electronic computers have received much attention. The various pseudo-languages represent great advances in this area, but a language is a great deal more than the basic tool of communication. A good language, -- a good symbology, -- is an essential element in man's thought processes. In a sense it defines his capacity for conceptualization and for abstract thought. It's no mystery that the telephone wasn't invented in Tahiti or the airplane in Afghanistan. Today we face a similar language restriction in trying to analyze and think about the complex decision-making systems required to operate a business or control an industrial process. Our traditional techniques seem inadequate. Flow charts quickly become a puzzle of lines, balloons, and boxes whose secret lies hidden in the mind of the creator. Frequently, programmers complain they would rather reprogram the job than take over someone else's flow charts.

In addition to flow charts, you often see matrix-type displays. They appear under a variety of names--collation charts, tabulated drawings, standard time data sheets, and so on. Often large and unwieldy, they usually represent listings of past decisions or answers rather than the logic used in making them. But none of these methods for thinking about and communicating complex decision logic have been really effective. Most business and professional people still communicate with the computer world through an elaborate hierarchy of flow charters and programmers. This is the problem which we feel TABSOL greatly simplifies. It combines some of the characteristics of earlier methods and introduces a few new features of tis own. After using TABSOL's decision structure tables in numerous applications, we feel they are both good communicating tools, and also valuable thinking tools. As one G.E. computer wag put it:

TABSOL is a thinking man's language.

Decision structure tables provide a standard, uniform methods for clearly describing complex, multi-variable multi-result decision systems.

A structure table consists of a rectangular array of terms, sub-divided into four quadrants. The vertical double line separates the decision logic on the left from the result functions or actions which appear on the right. A horizontal double line separates the structure table column headings above from the table values recorded in the horizontal rows below. Thus, the upper left quadrant records the names of the parameters effecting the decision while the lower left quadrant records the specific values which a decision parameter may have in a given situation. Similarly, the upper right hand quadrant records the names of result functions -- or actions to be performed -- once the decision has been made, and the lower right quadrant shows the actual result values which pertain directly opposite the appropriate set of decision parameter values. Thus, each horizontal row completely and independently describes one possible decision situation. Each structure table becomes a complete statement of the logical and quantitative relationships supporting a particular elementary decision.

There is no limit to the structure table columns or rows. The dimensions of any specific structure table are completely flexible, and are a logical consequence of the decision being described. A series of these structure tables taken in combination will describe a complete decision system.

Now let's look at a simple example (figure 1). Here we want to make an elementary decision on transportation from New York to Boston. There are three significant decision parameters: Weather, Plane Space, and Hotel Room. Weather has only two value states, Fair or Foul; Plane Space is either OK or Sorry; and Hotel Room can be either Open or Filled. In terms of results, Plane or Train are the only permissible means of Transportation. If the weather was Foul, despite an OK on plane space and an Open, Hotel Room, then we see by inspection that the solution appears in the second row. Train is the correct Transportation. We are also instructed to Cancel Plane, and this is the End of the decision.

This simple structure table provides a general solution to this particular decision-making problem. If afternoon trips to Boston ever occur -- and one must assume that they frequently do -- then an operating decision can quickly be made by supplying the current value of Weather, Plane Space, and Hotel Room and solving the structure table.

Solving a structure table consists of comparing or "testing" specific values assigned decision parameters in the problem statement against the corresponding sets of decision parameter values recorded in the structure table. If all tests in a row are satisfied, then the solution is in that row. The correct result values or actions appear in the same row, to the right of the double line.

Once a particular structure table has been solved and the result functions executed, it is often necessary to make more decisions. For this reason, the last result column of the structure table provides a firm link to the next decision structure table. Notice the last row specifies that for all values of Weather, with no Plane Space, and no Hotel Room, the decision-maker is directed to solve another structure table, Transportation, New York-Boston in the morning.

Similarly, a system designer can build a whole system of structure tables. He completely controls the make-up of each table, as well as its position in the sequence of total problem solution. He may decide to skip tables, or, he may re-solve tables to achieve the effect of iteration.

Getting from New York to Boston is a rather prosaic problem to say the least; we certainly don't need a computer to make decisions like this.

So let's look at how a systems designer might structure a real operating decision.

Table 2015 (figure 2) completely describes time standards determination for a certain coil winding operation. In this situation if the number of turns is less than 10, the operator's time allowance in seconds is equal to the number of turns. However, if the number of turns is greater than 10 but less than 15, the operator is allowed an additional 88 hundredths of a second.

So you see that problem values and decision parameter test values need not be simple identities. Actually, the problem values may be equal to the table value, greater than, less than, not equal to, greater than or equal to, less than or equal to the test value. This broad selection of test types greatly increases the power of individual structure tables and sharply reduces their size. Note that we can put the test type right in the test block immediately preceding the test value, or in the column heading after the decision parameter name. Of course, the test type in the column

heading applies to all test values appearing below. It is also possible to formulate complex test blocks involving two or more decision parameters.

Structure table results are not limited to simple assignments of alphabetic or numeric constants. As we've already seen if the solution occurs in the first row, the current value of `TURNS` is assigned `TIME`. If the solution occurs in the second row, the result of the arithmetic expression `TURNS / 0.88` is assigned `TIME`. If the solution occurs in the third or fourth row the result of the formula evaluation `TIME 1` or `TIME 2` will be assigned to `TIME`. This is the significance of the equal sign, appearing after the name in the result value block. These formulas are recorded in the area just prior to the structure table proper.

In the next action column the result function `PERFORM` appears. This means that the data processing or arithmetic operations named in the result value block are to be executed. Notice that one of the result values is another structure table. Should the solution occur in this row, Table 2016 will be solved just as any other, only control will remain within the framework of Table 2015 which is our illustrative table. When completed, the next result function will be processed. In the next column, the result function `GO` links this structure table to the next structure table to be solved. If there is no solution row found in the structure table proper, then control passes to the area directly below the structure table. This is usually regarded as an "error", and most often indicates a failure of the decision logic to cope with a certain combination of problem values. The systems designer can -- and should -- notify himself whenever such an error occurs by arranging for an error printout, identifying the table that failed and the problem being solved at the time. With this source language printout and other structure tables, the systems designer has all the data he needs to trouble-shoot the system in his own professional terminology.

We can also use the areas immediately before and after the structure table proper to record any additional language statements that may be required -- input output operations, data movement, operator instructions or any other data processing activities.

Of course, I cannot even attempt to completely describe decision structure tables in this short talk; a much more complete explanation appears in the Proceedings. There are many more features available for formulating concise, complete decision systems. I can only give a quick introduction to inherent Gestalt in this method of describing decision logic.

Structure tables did not start at their present state of development. This language concept evolved through a series of experimental tabular systems-oriented languages developed for the 305, 650, 702 and 704.

These experimental languages proved remarkably adequate; however, the added power of a conventional language seemed very appealing, particularly as the prospects for structure table application in all sorts of problem areas brightened.

At this point, General Electric's Computer Department joined the effort. The Computer Department had been developing a new compiler, called GECOM, for use with General Electric computers. The first version of this new General Compiler, will be available for the GE 225 in May, 1961. It has been designed primarily around COBOL, with some of the basic elements of ALGOL. It will now contain all of TABSOL. Simply stated, joining TABSOL with GECOM places the power of a full-fledged conventional language at the command of every structure table block.

We now have a rather substantial amount of experience in applying structure tables to a wide variety of operating decision-making problems. But perhaps the most interesting, at least from the researcher's point of view, was the very work which led to decision structure tables themselves. In 1957 we were investigating the possibility of automating the essential information and material processing required to directly transform customer orders into finished products. We studied customer order editing, product engineering, drafting, manufacturing methods, and time standards, quality control, cost accounting, and production control. This accounts for a fairly substantial portion of the operating decision system in a manufacturing business. Fortunately, the inputs and outputs to this system are simple and well-defined: the customer order comes in and the finished product goes out. So it was possible to treat all activities within these bounds as one integrated, goal-oriented operating decision system and develop decision structure tables accordingly. Working with a small product section in one of the Company's operating components, a significant portion of the functional decision logic was successfully structured. Then the resulting structure tables were directly incorporated into a computer-automated operating decision system which transformed customer orders for a wide variety of finished products directly into factory instructions for operators and numerically programmed machine tools. This prototype system was demonstrated to General Electric management in November, 1958. Since then, structure tables have been used to describe the operating decision logic in many different applications. Structure tables appear to have great potential in compilers and also in computer simulation programs.

Problem Statement: Select Transportation, New York - Boston, p. m.

Weather: Foul

Plane Space: OK

Hotel Room: Open

Decision Structure Table: Transportation, New York - Boston, p. m.

Weather	Plane Space	Hotel Room	Trans- portation	Other In- structions	Next Decision
Fair	OK	Open	Plane		End
Foul	OK	Open	Train	Cancel Plane	End
	Sorry	Open	Train		End
	OK	Filled		Cancel Plane	NY-Bost. a. m.
	Sorry	Filled			NY-Bost. a. m.

Solution:

If the value of Weather is Foul, and
the value of Plane Space is OK, and
the value of Hotel Room is Open,

Then

the value of Transportation is Train, and
the value of Other Instructions is Cancel Plane, and
the value of Next Decision is End.

Figure 1

TABLE 2015. DIMENSION C2 A3 R4.

NOTE TIME STANDARDS FOR COIL WINDING

TIME ~1 = 125*DIA*TURNS.

TIME ~2 = 1000*DIA/SQRT (TURNS).

BEGIN

URNS	URNS LS	TIME	PERFORM	GO
LS 10		TURNS		TABLE 2020
GREQ 10	15	TURNS + 0.88	SETUP	TABLE 2025
GREQ 15	100	TIME ~ 1 =	SETUP	TABLE 2025
GREQ 100	1000	TIME ~ 2 =	TABLE 2016	TABLE 2030

IF NOT SOLVED GO ERROR ~COIL.

END TABLE 2015.

IBM Corporation
Research Laboratory
P. O. Box 218
Yorktown Heights, New York

February 15, 1961

Memorandum to:


Subject: Tabular Techniques Development
 Distribution #2

This is the second release of material concerning the development of tabular techniques for systems and programming description. Enclosed are two items:

- (1) A working paper by Mr. Earl Althoff of Eastman-Kodak describing a tabular approach to a file updating problem.
- (2) A preliminary report on TABSOL 225 by Mr. D. Klick of General Electric's Computer Department. This paper was given at the CODASYL Systems Committee meeting in December, 1960.

Reference is also made to a third item which is available through IBM Stationary Stores in Endicott and hence not attached:

- (3) General Information Manual "Advanced Analysis Method for Integrated Electronic Data Processing" by Mr. Orren Y. Evans of Hunt Foods & Industries. This is Report No. F20-8047.


Burton Grad
Project Coordinator



CLEARINGHOUSE REPORT

A PRELIMINARY APPROACH
TO
TABULAR PROGRAMMING

February 1, 1961
Ref. No. 1B1

E. O. Althoff
Data Processing Service
Eastman - Kodak

INTERNATIONAL BUSINESS MACHINES CORPORATION

White Plains, New York

This material is distributed to keep IBM personnel informed of new developments. Selection is based on interest; this department makes no claim for the desirability of this approach nor necessarily recommends its use.

If additional copies are desired, please contact the Clearinghouse. No part of this material should be reproduced or distributed outside IBM without approval of the Clearinghouse.

POINTS ABOUT PRELIMINARY APPROACH TO TABULAR PROGRAMMING

1. For each element used, prepare a 15-digit title to use in the English text and a four-digit abbreviation to use in formulae. The four-digit abbreviation either starts with a letter or is numbered sequentially 0001, 0002, 0003,

2. Do not strain to over-abbreviate. For example, CTO1, CTO2, ... can be used to stand for control totals of various types. It is usually best to give mnemonic abbreviations only for the hundred or less most used elements.

3. Data sets can be listed on a data element sheet if desired. For example, the data set "Target Date" abbreviated TRGT consists of the data elements:

"Target Month" abbreviated TMON
"Target Day" abbreviated TDAY
"Target Year" abbreviated TYR

In the above, four entries are made, one for each data element and one for the data set.

4. The definition should be clear and unambiguous, but above all must be complete. Differentiate clearly between similar data elements.

5. Prepare a data file for each set of data (not going directly to a report). Do not consider the machine in your preparation. As an example consider a tape with records of Type A followed by several records of Type B; prepare two data files, A and B, since having these on the same tape is pure machine method.

6. For each data set or element listed, record a reference to set number and page number of the data element sheets. Thus, 03-01 refers to data element set 03, page 01. Record both the title and the abbreviation. Record the length for that file. A given element can require four digits on one data file and six on another.

7. Give each data file a letter designation A, B, C, .., record whether input or output. In the case of an updated data file, assign two letters, say A for input and B for output.

8. Obtain a Data Processing Spacing Chart for all report lines (messages as well as fancy reports). Label each report A, B, etc. Use a second letter for each different type of line. Thus, Report A may have lines AA, AB, AC, ...

9. Tables must give all logic except how to start and how to stop. All the statements which follow must be accomplished completely — no exceptions can be permitted.

10. The table is divided into conditions and actions. On the left, one gives the English statement of the condition or action, and on the right, one records the precise formula fully and completely. Thus, on the right;

A-STAT = B-STAT clearly shows that the condition is true if and only if the STAT element of data File A equals the STAT element of data File B.

C-0001 > 0 shows the condition is true only if the data element 0001 of data File C is greater than the constant 0.

11. The formula for a condition can include any connectors desired to complete "a single condition". Examples are:

F-TAX = 10 or 15
F-TAX = 10 and G-TAX = 15

12. The actions can be varied also. In general, one records data movement or arithmetic actions first, then all data file advance actions, then all table transfer actions.

13. Typical data movement actions are D → E (ASG#, PROG, and TT01) meaning move from data File D to data File E the data elements ASG#, PROG, and TT01. In case of one move, D-ASG# → E-ASG#. Others are D-PDHR add to E-YRHR, etc.

14. When data is posted to report lines, increment is used as: D-ASG# → AB14 meaning post data element ASG# of data File D to position 14 (right-hand increment) of line B of Report A.

15. Another action may be to do an action or actions from other tables. Thus, Action 2 of Table 01-A5 can simply be "Do actions 3 and 6 of Table 03-B7.

16. Another action may follow the actions for a data rule from another table in its entirety; if so, simply transfer to Rule XX of Table XX-XX.

17. The advance data files actions are abbreviated GIV X for input and TAK X for output. In some cases posting a data element to a control total is included as: C-AMT add to TOT1; TAK C. When an advance action is given, the next action calling on that file from any table will be from the next record.

18. Tables are numbered NN-XN where NN denotes the project area; NN runs from 01, 02, ., X is a letter denoting a sub-project and runs from A to Z, while the rightmost N is 1 to 9 and denotes table within sub-project. The last action for any data rule is always a transfer to some table. (Do not transfer to a rule within a table — leave this to the programmer,)

19. On any given table, possible entries opposite condition are Y, N, and -. Y = Yes, N = No, and - means "does not apply".

The matrix

Y	Y	N
Y	N	N

 would indicate an analyst omission since the combination

N
Y

is not specified. One must specify enough data rules to account for every combination of the conditions, whether possible or impossible (is it really impossible???)

The - is used primarily in two cases:

A. If condition 1 is A-STAT = 0 and condition 2 is A-STAT = 1, then a

Y	-
-	Y

 entry would show that, if A-STAT = 0, we don't need to test for A-STAT = 1 and vice versa.

B. The - may be used to indicate a plain transfer to another table, when the only alternative would be to over-run the 6 X 10 matrix. Example:

Condition 1	Y	Y	Y	Y	Y	Y	Y	Y	N
Condition 2	Y	Y	Y	N	Y	N	N	N	-
Condition 3	Y	Y	N	Y	N	Y	N	N	-
Condition 4	Y	N	Y	Y	N	N	Y	N	-

The data rule on the right simply transfers to another table where the NO's for Condition 1 are spelled out.

20. In summary, the preliminary approach is designed to obtain from a job analyst actions to follow for every combination of conditions. The conditions and actions are not to be vague — but must be 100 percent precise to every data element involved. There is no thought given in the preliminary approach to automating any of the steps: tables → programs. Only a person with two - three years active programming and computer systems experience can prepare tables containing many subtle traps which develop only in automatic E.D.P. systems; for the next year or so, it is expected that these people will return expanded tables (with these subtle points included) to the job analyst and will, in addition, write programs in KodaKoder.

ELEMENTS DEFINITION

Name: Earl O. Althoff

SET # 01 PAGE #01

Project: D.P.S. Billing1-DIGIT TITLE One Letter ASGN4-DIGIT ABBREV. ASGL

DEFINITION: A letter code used to differentiate between several types of perpetual assignments. Refer to the back of a D.P.S. Time-Reporting Sheet for full details.

15-DIGIT TITLE Assignment No.4-DIGIT ABBREV. ASG#

DEFINITION: A four-digit number given in sequence to non-perpetual assignments as they occur. The number has no structure of any sort.

15-DIGIT TITLE Billing Number4-DIGIT ABBREV. BIL#

DEFINITION: A five-digit number assigned by the D.P.S. Accountant to each account or sub-account which D.P.S. bills. It is structured as desired to yield a meaningful report order.

15-DIGIT TITLE PROG-SYSTEM NO. (DATA SET)4-DIGIT ABBREV. UJ#

DEFINITION: A uniform job number which serves a variety of purposes. It is organized primarily by computer run and program within computer run. (See Chapter 27 - Self-Teach.) Consists of elements MFC, RUN#, and PRGL.

15-DIGIT TITLE Project Title4-DIGIT ABBREV. TITL

DEFINITION: A 45-character title given to each project having a four-digit assignment number.

15-DIGIT TITLE Project Type Code4-DIGIT ABBREV. TYPE

DEFINITION: A two-character code enabling us to group a project by new programs (N), changes (C), or revision (R). The units position is 1 for a business project, 6 for a program research project.

15-DIGIT TITLE Major Fctn. Code4-DIGIT ABBREV. MFC

DEFINITION: A two-digit code used by D.P.S. to roughly distinguish between basic major project functions such as Merchandise Billing, Paper Finishing Scheduling, etc. It is the first two digits of Prog-System No.

6-DIGIT TITLE Target Date4-DIGIT ABBREV. TRGT

DEFINITION: The date by which an assignment should be completed to the point that production results are obtainable. Six digits as 011560.

DATA PROCESSING SERVICE

ELEMENTS DEFINITION

SET # 01 PAGE # 02

Job No.: _____

Name: Earl Althoff

Project: D.P.S. Billing

15-DIGIT TITLE Programmer 4-DIGIT ABBREV. PROG

DEFINITION: An official ten-digit name assigned to each individual of the Programming and Methods Staff

15-DIGIT TITLE Registration # 4-DIGIT ABBREV. REG#

DEFINITION: A six-digit number given to each employee of Kodak Rochester. The first three digits indicate department and the last three are sequentially given by various rules.

15-DIGIT TITLE Prog-Syst.Descr. 4-DIGIT ABBREV. DESC

DEFINITION: Refers to a 29-digit alphanumeric title or description given to each specific program or computer systems sub-assignment.

15-DIGIT TITLE EST. Man Months 4-DIGIT ABBREV. ESTM

DEFINITION: Refers to a time estimate given for each program in an assignment. The time is given in four digits (one decimal place).

15-DIGIT TITLE Due Date for V 4-DIGIT ABBREV. DUEV

DEFINITION: A date given for each program to be ready for system volume testing. Six digits as 011560 or 12B161.

15-DIGIT TITLE Department 4-DIGIT ABBREV. DEPT

DEFINITION: A four-character alphanumeric abbreviation of the department a programmer belongs to. Examples are DPS, MSDD, A&O, DC.

15-DIGIT TITLE Computer Run # 4-DIGIT ABBREV. RUN#

DEFINITION: The third and fourth digit of Prog-System No.. Delineates the programs constituting a scheduled computer run.

5-DIGIT TITLE Program Letter 4-DIGIT ABBREV. PRGL

DEFINITION: The fifth digit of Prog-System No. Letters from A to Z are given to programs of a given computer run.

DATA PROCESSING SERVICE
DATA FILE LAYOUT

Job No.: _____

Project: D.P.S. Billing

FILE DESCRIPTION Assignment Master

Data File: A in B out

Name: Earl Althoff

TITLE	REF.	ABBREV.	LENGTH	For Programmer Use Onl	
				REC #	INCR.
1. Assignment No.	01-01	ASC#	4	_____	_____
2. Billing Number	01-01	BIL#	5	_____	_____
3. Proj. Ldr. Name	01-05	PLDR	10	_____	_____
4. Project Title	01-01	TITL	45	_____	_____
5. Proj. Type Code	01-01	TYPE	2	_____	_____
6. Major FCTN Code	01-01	MFC	2	_____	_____
7. Target Date	01-01	TRGT	6	_____	_____
8. Bill-out Code	01-01	BILC	1	_____	_____
9. Completion Code	01-03	CMPL	1	_____	_____
10.	_____	_____	_____	_____	_____
11.	_____	_____	_____	_____	_____
12.	_____	_____	_____	_____	_____
13.	_____	_____	_____	_____	_____
14.	_____	_____	_____	_____	_____
15.	_____	_____	_____	_____	_____
16.	_____	_____	_____	_____	_____
17.	_____	_____	_____	_____	_____
18.	_____	_____	_____	_____	_____
19.	_____	_____	_____	_____	_____
20.	_____	_____	_____	_____	_____
21.	_____	_____	_____	_____	_____
22.	_____	_____	_____	_____	_____
23.	_____	_____	_____	_____	_____
24.	_____	_____	_____	_____	_____
25.	_____	_____	_____	_____	_____
26.	_____	_____	_____	_____	_____
27.	_____	_____	_____	_____	_____
28.	_____	_____	_____	_____	_____
29.	_____	_____	_____	_____	_____
30.	_____	_____	_____	_____	_____
31.	_____	_____	_____	_____	_____
32.	_____	_____	_____	_____	_____
33.	_____	_____	_____	_____	_____
34.	_____	_____	_____	_____	_____
35.	_____	_____	_____	_____	_____

DATA PROCESSING SERVICE
DATA FILE LAYOUT

Job No.: _____

Project: D.P.S. Billing

FILE DESCRIPTION Current Time Records

Data File: C in

Name: Earl Althoff

TITLE	REF.	ABBREV.	LENGTH	For Programmer Use Only	
				REC #	INCR.
1. Assignment No.	01-01	ASG#	4	_____	_____
2. Prog-System No.	01-01	UI#	5	_____	_____
3. Programmer	01-02	PROG	10	_____	_____
4. Department	01-02	DEPT	4	_____	_____
5. Progress Code	01-03	STAT	1	_____	_____
6. Est. Date for V	01-03	ESTV	6	_____	_____
7. Hrs. This Period	01-03	HRTF	4	_____	_____
8. CPU MIN V-Test	01-05	VTST	4	_____	_____
9. K-10S This Pd.	01-04	K10P	2	_____	_____
10. K-20S This Pd.	01-04	K20P	2	_____	_____
11. K-30S This Pd.	01-03	K30P	2	_____	_____
12. K-40S This Pd.	01-04	K40P	2	_____	_____
13. K-50S This Pd.	01-04	K50P	2	_____	_____
14. ASGL can be M and N only	01-01	ASGL	1	_____	_____
15. _____	_____	_____	_____	_____	_____
16. _____	_____	_____	_____	_____	_____
17. _____	_____	_____	_____	_____	_____
18. _____	_____	_____	_____	_____	_____
19. _____	_____	_____	_____	_____	_____
20. _____	_____	_____	_____	_____	_____
21. _____	_____	_____	_____	_____	_____
22. _____	_____	_____	_____	_____	_____
23. _____	_____	_____	_____	_____	_____
24. _____	_____	_____	_____	_____	_____
25. _____	_____	_____	_____	_____	_____
26. _____	_____	_____	_____	_____	_____
27. _____	_____	_____	_____	_____	_____
28. _____	_____	_____	_____	_____	_____
29. _____	_____	_____	_____	_____	_____
30. _____	_____	_____	_____	_____	_____
31. _____	_____	_____	_____	_____	_____
32. _____	_____	_____	_____	_____	_____
33. _____	_____	_____	_____	_____	_____
34. _____	_____	_____	_____	_____	_____
35. _____	_____	_____	_____	_____	_____

DATA PROCESSING SERVICE

DATA FILE LAYOUT

FILE DESCRIPTION Program System Master

Job No.: _____

Project: D.P.S. Billing

Data File: K in L out

Name: Earl Althoff

TITLE	REF.	ABBREV.	LENGTH	For Programmer Use Onl	
				REC #	INCR.
1. Assignment No.	01-01	ASG#	4	_____	_____
2. Prog-System No.	01-01	UJ#	5	_____	_____
3. Programmer	01-02	PROG	10	_____	_____
4. Prog-System Description	01-02	DESC	29	_____	_____
5. Est. Man Months	01-02	ESTM	4	_____	_____
6. Due Date for V	01-02	DUEV	6	_____	_____
7. Department	01-02	DEPT	4	_____	_____
8. Progress Code	01-03	STAT	1	_____	_____
9. Est. Date for V	01-03	ESTV	6	_____	_____
10. HRS to Date	01-03	HRTD	5	_____	_____
11. Bill-out Code	01-03	BILC	1	_____	_____
12. V-TEST To Date	01-05	VFTD	6	_____	_____
13. K-10S To Date	01-04	K10S	3	_____	_____
14. K-20S To Date	01-05	K20S	3	_____	_____
K-30S to Date	01-04	K30S	3	_____	_____
16. K-40S To Date	01-04	K40S	3	_____	_____
17. K-50S To Date	01-04	K50S	3	_____	_____
18. ASGL will be M and N only	01-01	ASGL	1	_____	_____
19.	_____	_____	_____	_____	_____
20.	_____	_____	_____	_____	_____
21.	_____	_____	_____	_____	_____
22.	_____	_____	_____	_____	_____
23.	_____	_____	_____	_____	_____
24.	_____	_____	_____	_____	_____
25.	_____	_____	_____	_____	_____
26.	_____	_____	_____	_____	_____
27.	_____	_____	_____	_____	_____
28.	_____	_____	_____	_____	_____
29.	_____	_____	_____	_____	_____
30.	_____	_____	_____	_____	_____
31.	_____	_____	_____	_____	_____
33.	_____	_____	_____	_____	_____
34.	_____	_____	_____	_____	_____
35.	_____	_____	_____	_____	_____

DATA PROCESSING SERVICE

ELEMENTS DEFINITION

Job No.: _____

Name: _____

Project: _____

SET # PAGE #

15-DIGIT TITLE _____ 4-DIGIT ABBREV. _____

DEFINITION: _____

15-DIGIT TITLE _____ 4-DIGIT ABBREV. _____

DEFINITION: _____

15-DIGIT TITLE _____ 4-DIGIT ABBREV. _____

DEFINITION: _____

15-DIGIT TITLE _____ 4-DIGIT ABBREV. _____

DEFINITION: _____

15-DIGIT TITLE _____ 4-DIGIT ABBREV. _____

DEFINITION: _____

15-DIGIT TITLE _____ 4-DIGIT ABBREV. _____

DEFINITION: _____

15-DIGIT TITLE _____ 4-DIGIT ABBREV. _____

DEFINITION: _____

5-DIGIT TITLE _____ 4-DIGIT ABBREV. _____

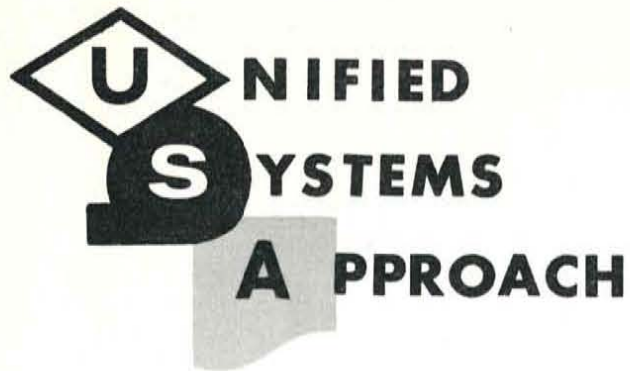
DEFINITION: _____

DATA PROCESSING SERVICE
DATA FILE LAYOUT

Job No.: _____
Project: _____
Data File: _____
Name: _____

FILE DESCRIPTION _____

	<u>TITLE</u>	<u>REF.</u>	<u>ABBREV.</u>	<u>LENGTH</u>	<u>For Programmer Use Onl</u>	
					<u>REC #</u>	<u>INCR.</u>
1.	_____	_____	_____	_____	_____	
2.	_____	_____	_____	_____	_____	
3.	_____	_____	_____	_____	_____	
4.	_____	_____	_____	_____	_____	
5.	_____	_____	_____	_____	_____	
6.	_____	_____	_____	_____	_____	
7.	_____	_____	_____	_____	_____	
8.	_____	_____	_____	_____	_____	
9.	_____	_____	_____	_____	_____	
10.	_____	_____	_____	_____	_____	
11.	_____	_____	_____	_____	_____	
12.	_____	_____	_____	_____	_____	
13.	_____	_____	_____	_____	_____	
14.	_____	_____	_____	_____	_____	
15.	_____	_____	_____	_____	_____	
16.	_____	_____	_____	_____	_____	
17.	_____	_____	_____	_____	_____	
18.	_____	_____	_____	_____	_____	
19.	_____	_____	_____	_____	_____	
20.	_____	_____	_____	_____	_____	
21.	_____	_____	_____	_____	_____	
22.	_____	_____	_____	_____	_____	
23.	_____	_____	_____	_____	_____	
24.	_____	_____	_____	_____	_____	
25.	_____	_____	_____	_____	_____	
26.	_____	_____	_____	_____	_____	
27.	_____	_____	_____	_____	_____	
28.	_____	_____	_____	_____	_____	
29.	_____	_____	_____	_____	_____	
30.	_____	_____	_____	_____	_____	
31.	_____	_____	_____	_____	_____	
32.	_____	_____	_____	_____	_____	
33.	_____	_____	_____	_____	_____	
34.	_____	_____	_____	_____	_____	
35.	_____	_____	_____	_____	_____	



CLEARINGHOUSE REPORT

PRELIMINARY REFERENCE MANUAL DRAFT

TABSOL - 225

A Tabular Systems Oriented Language
for the
GE 225 Information Processing System

February 1, 1961
Ref. No. 1B2

D. Klick
Computer Department
General Electric Company

INTERNATIONAL BUSINESS MACHINES CORPORATION

White Plains, New York

This material is distributed to keep IBM personnel informed of new developments. Selection is based on interest; this department makes no claim for the desirability of this approach nor necessarily recommends its use.

If additional copies are desired, please contact the Clearinghouse. No part of this material should be reproduced or distributed outside IBM without approval of the Clearinghouse.

PRELIMINARY REFERENCE MANUAL

TABSOL-225 --- A Tabular Systems Oriented Language

for the

GE225 Information Processing System

**Computer Department
Applications Section
Programming Research and Development
Phoenix, Arizona**

December, 1960

This document is a draft of a Preliminary Reference Manual and a language specification for integrating decision tables with the General Compiler. The information contained herein assumes a basic knowledge of computers and electronic data processing applications. Therefore, the manual should be used not as a text book but rather to augment already realized skills. Minor changes in language specification may occur during the implementation period of the compiler. Any changes that are made will be reflected in future and more final versions of this manual or in supporting material issued during the interim of implementation.

D. Klick
Programming Research

I. INTRODUCTION

Early automatic coding systems, such as assembly programs, employed mnemonic abbreviations in place of the computer's numerical instruction code and symbolic addresses in place of actual memory addresses. In reality the assembly program language was a set of synthetic computer instructions. Although these systems greatly simplified programming, the programmer was still plagued with the many details dictated by the computer.

Automatic coding languages of today are on the threshold of relieving the programmer of these details. The structure of these new languages are very much like English. By using a combination of English words and phrases to form sentences, the programmer now needs only to "describe" a procedure for the computer to follow. This procedure together with a description of the data is then given to a special computer program for processing. This special program, commonly called a compiler, translates the English problem description and generates a program of computer instructions.

Such a compiler is provided for the GE-225. Its General Compiler evolved from two noteworthy language efforts - the Common Business Oriented Language (COBOL) and the Algorithmic Language (ALGOL). Both languages were developed by voluntary committees of computer manufacturers and users and reflect the recent trend toward "common" compiler languages.

The language presently available with the General Compiler is based primarily on COBOL, since COBOL satisfied the needs of a broad spectrum of data processing applications. To accommodate the demands of more technical

applications, Boolean expressions, floating point arithmetic, and the ability to express equations were incorporated into the format of COBOL. Therefore, one may say that the present version of the General Compiler can accept programs written in one, two, or in a combination of two language forms.

Those programmers familiar with COBOL recognize that it is well suited for creating and processing data files. ALGOL, on the other hand, provides an excellent means for expressing complex mathematical relationships. Recent investigations by the Integrated Systems Project of General Electric's Manufacturing Services uncovered an area of applications which require neither extensive data file processing nor profound mathematics but rather an unwieldy number of sequential decisions.

To cope effectively with these decisions the ISP team devised a tabular language. The purpose of this language was to depict, by means of tables, the relationships of logical decisions. The new language was appropriately termed TABSOL for Tabular Systems Oriented Language. Since its creation TABSOL has been used by many departments of General Electric to analyze and solve problems in product engineering, manufacturing methods, cost accounting, and production control. The application of decision tables is continually growing. Recent studies show that they provide a concise method for supporting the logic of other data processing applications. For example, decision tables may be used to specify the transfer of control associated with the values of one or more fields, to control the printing of detail and summary lines of a report, or to interrogate the sort keys in a multi-file system. At

the Computer Department we have found decision tables a valuable tool in designing and implementing the General Compiler.

Decision tables represent a third language for the General Compiler. They may be used by themselves or in conjunction with the features of the compiler language. The specifications outlined in this manual pertain mainly to the table entries and imply and require a knowledge of the General Compiler. Therefore, this manual should be used as a supplement to the GZ-225 General Compiler Manual, CPE-123 (5.5M10-60).

II. DECISION TABLE FORMAT

The format of a decision table is given in Fig. 1. In concept a table is an array of blocks divided into four quadrants by a pair of double lines. The vertical double line separates the decisions or "conditions" on the left from the "actions" on the right. The horizontal double line isolates variables from associated operands which will appear in the blocks and rows below. A condition then is a relation between a variable appearing in a primary block and an operand appearing in a corresponding secondary block. For example, we may write AGE in primary block 1 and EQ 26 in secondary block 1. In doing this, we are stating a condition. Verbally, we are asking "if age equals 26". An action, on the other hand, is a statement of what is to be done. By writing AGE in a primary action block and 26 in its associated secondary block, we are stating that "the value 26 is to be assigned to age".

It is interesting to note, at this point, the English interpretation given to the vertical lines. The left-most line may be thought of as representing the word IF. Those lines to the left of the vertical double line may be taken to mean AND; the vertical double line itself the word THEN. Since actions are sequential entities, the lines separating them may be interpreted as semicolons and the right-most line, which actually terminates the actions, as a period. With this in mind, each secondary row becomes an English sentence. For example, each row now reads:

"IF condition-1 is satisfied AND condition-2 is satisfied
AND . . . AND condition-k is satisfied THEN perform
action-1; action-2; . . . ; action m."

If any condition within a row is not satisfied, the next row is evaluated

and so on until all the rows are depleted. When this happens the table is said to have "no solution". The table is considered "solved" when all the conditions of a row are satisfied and their associated actions performed.

Before considering the conventions used to formulate conditions and actions, an example may help develop insight into the nature of decision tables and the manner in which they may be used with the General Compiler. In this example (Fig.2) we are searching a master employee file (recorded on magnetic tape) to determine the number of male employees who fall into the following job categories.

<u>Job Level</u>	<u>Years Experience</u>	<u>Title</u>
6	2	Programmer
7	3	Programmer or Analyst
8	More than 3	Analyst
9	More than 4	Analyst or Manager
10	More than 4	Manager

For each employee we find having any of these qualifications, we are to write his department number, name, title, level, and experience on the computer's typewriter. At the end of the run the totals for each of the categories are to be also put on the typewriter.

The core of this problem is the decisions that must be made on the information stored in the records of the master file. These decisions are conveniently expressed above in narrative form. With only minor alteration this form becomes the program statement of our problem. The table and sentences are punched into 80-column cards exactly as they appear in Fig.2. When this is done they may be given directly to the compiler for processing.

As illustrated in our example, General Compiler sentences may be used to support the logic of the table. These sentences accomplish the following:

OPEN --- Declares that the **MASTER-FILE** is input and since the file is recorded on magnetic tape, validates the tape labels.

READ --- Delivers the next record from the **MASTER-FILE** and tests for an end-of-file sentinel. When this sentinel is detected, sequential program execution is interrupted and control passes to the portion of the program labeled **END-RUN**.

IF --- Eliminates those data records which contain information about female employees. The word **FEMALE** (also **PROGRAMMER**, **ANALYST**, and **MANAGER** used in the table) represents a special kind of condition and will be explained later in the manual.

EXPERIENCE = --- Calculates the employees total experience and assigns the value to the field named **EXPERIENCE**.

The word **TABLE** informs the compiler that it must process a decision table; **EXAMPLE** is a name or label which was given to the table. The size of the table is stated next by giving the number of conditions, actions, and rows contained in the table. This information is used only by the compiler and is not executed by the compiled program.

Table execution begins at row 1 (sequence number 40). Using our narrative definition of a table, row 1 is interpreted as follows:

"IF the job **LEVEL** field equals (EQ) 6 AND the **EXPERIENCE** field equals (EQ) 2 years AND the employee's title is **PROGRAMMER** THEN assign the value 1 to the subscript **I**; GO TO the part of the program having the label **TYPE-OUT**."

If one of these conditions cannot be satisfied, row 2 is evaluated starting again with the left-most condition. Sequential execution of the rows continues until either all conditions in a given row are satisfied or

all rows are exhausted. When the latter situation occurs, the sentence immediately following the table is executed. Proceeding from here the sentences in our example accomplish the following:

GO --- Interrupts sequential program execution and passes control to the part of the program labeled **GET-RECORD**.

WRITE--- Writes the current contents of the **DEPARTMENT**, **NAME**, **TITLE**, **LEVEL**, and **EXPERIENCE** fields on the computer's typewriter.

CLOSE--- Rewinds the **MASTER-FILE** and performs the file's closing conventions.

STOP --- Terminates processing and writes the words **END RUN** on the typewriter.

By General Compiler standards this example represents relatively simple conditions and actions. In formulating these entries, the programmer may take full advantage of the compiler's capabilities. The remaining sections of this manual are devoted to defining the conventions and manner in which conditions and actions may be formed and entered in tables.

III. BASIC CONCEPTS

Since decision tables are used in conjunction with the General Compiler language, we must first look at the foundations of this language before considering the counterparts that may appear in a table. The compiler's language, like most natural languages, is a body of words and a set of conventions for combining these words to express meanings. Its structure or "syntax" closely resembles the rules of English grammar, and its body of words may be appropriately termed a "vocabulary". The purpose of this section is to show how words are formed and how they may be used to express a desired meaning.

Characters

The basic units of our language are the characters used to form words and symbols. The character set includes the letters of the alphabet (A, B, C, ..., Z), the numerals (0, 1, 2, ..., 9), and the special characters shown in Fig. 3. Special characters are presented in more detail as they are encountered in the manual.

Words

The words of a typical General Compiler program fall into one of two categories: the vocabulary of the compiler and the vocabulary used by the programmer. The programmer's vocabulary will consist mostly of arbitrary names given to his data and sections of his program. The compiler's vocabulary, on the other hand, is predetermined and explicitly defined in this manual. Since the compiler, by nature of its designers, is a mistrusting mechanism, the programmer must define the words he uses too. This is done, not by writing a manual, but instead by merely

SPECIAL CHARACTERS

<u>Character</u>	<u>Meaning</u>	<u>Card Code</u>
Δ	Space or blank	Space
.	Period - Decimal point	12-3-8
,	Comma	0-3-8
"	Quotation Mark	3-8
~	Hyphen	5-8
(Left Parenthesis	0-5-8
)	Right Parenthesis	0-6-8
+	Addition	12
-	Subtraction - Minus Sign	11
*	Multiplication	11-4-8
/	Division	0-1
=	Assignment	6-8
	Vertical Table Line	12-4-8

Figure 3

filling out a data description form. Once these "data names" are defined, they may be filed either on 80-column punched cards or on magnetic tape and used over and over again. The data description file then is a "dictionary" since it contains the definitions of the words used by the programmer. Furthermore, this dictionary may be revised without redefining all of its entries. This is accomplished by a special service routine which accepts corrections, insertions, and deletions as long as they are written on the compiler's data description form.

Our two categories of words may be illustrated by the following sentence taken from the program example given in Fig. 2.

GET-RECORD. READ MASTER-FILE RECORD IF END FILE GO TO END-RUN.

Here, the words READ, RECORD, IF, END, FILE, GO, and TO belong to the vocabulary of the compiler; whereas, the words GET-RECORD, MASTER-FILE, and END-RUN belong to the programmer's vocabulary. The compiler will assume that MASTER-FILE is a data name due to the word's position in the sentence. It will then search the data description to verify its assumption and to determine the characteristics depicted by this word. Not finding a match in the data description results in an error message typed on the computer's typewriter. The words GET-RECORD and END-RUN will be interpreted as sentence names due to their position in the program. Once again, the compiler will attempt to verify its findings by checking each transfer to make certain that they lead to properly defined sentence names. The consequence of an undefined sentence name is likewise an error message on the computer's typewriter. The compatibility checks mentioned here are only two of many which the compiler performs to insure unquestionable results in the programs which it creates.

Formation of Names

As previously mentioned, data names are words representing data (files, records, fields, elements, constants, arrays of values, etc.) and are arbitrarily assigned by the programmer. They are formed from the following characters.

Letters	A, B, C, ..., Z
Numerals	0, 1, 2, ..., 9
Hyphen	-

To avoid error messages and possible re-compilation, the programmer should choose data names that

1. Do not exceed 12 characters,
2. Do contain at least one letter,
3. Do not begin or end with a hyphen.

To insure a properly defined program, all data names should be recorded and their characteristic data described on the compiler's data description form. The programmer also should be careful not to use the compiler's vocabulary as data names.

In addition to data names, the programmer is free to name sentences, tables, and other "procedures" in his program. With one exception these names are formed like data names. Since procedure names are judged from their position in the program, they may be formed from only the numerals, 0 through 9.

Constants

The values associated with data names generally change during the actual running of a compiled program. It is for this reason that they

are sometimes called "variables". A constant, as opposed to a variable, is a specific value and does not change within the scope of a program. Constants may be one of two kinds: a literal, or a named constant.

A literal is a value itself rather than a name given to a value. Literals may be numerical, alphabetic, or alphanumeric - i.e., composed from the character set of the computer. All non-numeric literals should be enclosed in quotation marks (") to avoid having the compiler confuse them with data names. The conventions for forming literals are the following:

1. Non-numeric literals are limited to 30 characters, excluding the quotation marks.
2. A numeric literal not enclosed in quotation marks is assumed to be a number. Numbers may contain not more than one decimal point and a minus sign. Unsigned numbers are considered positive. Excluding decimal points and minus signs, numbers must not exceed 11 decimal digits.
3. Numbers may be treated as floating point by writing them as a power of ten - i.e., a number or decimal fraction followed by a power of ten exponent. For example, the number 230100 might be written as 2.301E5 which is equivalent to 2.301 multiplied by 10^5 . The exponent part, indicated by the letter E, may contain a minus sign to show a negative exponent. The value range of an exponent is limited to ± 75 . Excluding the decimal point, the minus sign, and

the letter E, the fractional part of a power of ten number must not exceed nine decimal digits. To distinguish data names from floating point numbers, data names should not be formed from only the numerals and the letter E.

4. An alphanumeric literal may not contain an embedded quotation mark since the enclosing quotation marks are used to determine the size and content of the literal.

A named constant is a constant which has been given a name. Named constants are defined by means of the data description and may include any character belonging to the character set of the computer, including the quotation mark. Like literals named constants may be numeric, alphabetic, or alphanumeric. They are unlike literals in that they may be any length.

Subscripts

Subscripts provide a convenient method to reference individual values contained in a list or in an array of values. The variable, I, employed in the decision table of Fig. 2 is a subscript used just for this purpose. Since five totals are to be accumulated, one name was assigned to all five, namely, the data name TOTAL. Whenever reference was made to a particular total, the data name TOTAL was followed by the subscript I. This is illustrated in the expression

$$\text{TOTAL (I)} = \text{TOTAL (I)} + 1.$$

and the sentence which prints all five totals on the typewriter. From this example, it follows that subscripts, like data, may be given names. In fact the same rules that govern forming data names apply to naming subscripts.

Since subscripting is a positional notation, the range of any subscript is limited to the values 1, 2, 3, . . . , n (where n is the maximum number of values in a list). This does not mean that subscripts are limited only to integers. If a subscript is not defined as integer by means of the data division, the compiler will automatically provide coding to truncate its value to an integer. Furthermore, subscripts are not restricted to a single variable name. Arithmetic expressions may also be used as subscript. For example,

```
RATE (P+1)
K ((X-3)*P**3)
A (J)
```

are legitimate forms of subscripts.

Up until now, only one-dimensional subscripting was considered. Values in multi-dimensional arrays may also be referenced by subscripts. For example, an array in which values are ordered

```
A11 A12 A13 A14 A15
A21 A22 A23 A24 A25
A31 A32 A33 A34 A35
A41 A42 A43 A44 A45
A51 A52 A53 A54 A55
```

might be subscripted as A (J,K), where K is the columnar subscript and J the row. To refer to value A₃₅, J would have to equal 3 and K equal 5.

Preceding examples show that subscripts are enclosed in parenthesis and separated by commas. This notation permits the compiler to distinguish subscripts from other elements in the language.

Truth-Values

There is a class of variables which, through either usage or definition, may assume only the numerals 1 or 0. The value 1 is said to be their true state and the value 0 their false state. The words END FILE of the READ sentence in Fig. 2 is such a variable. When the OPEN sentence is executed, END FILE is set to its false state and remains so set until the end-file condition is encountered. At this time, it is set to its true state.

Variables having truth-values are termed "True-False" variables. END FILE is a convenience provided by the compiler; the programmer may also formulate his own true-false variables by merely listing them under the heading TRUE/FALSE in the data division. They may be named according to the rules given for data names.

Arithmetic Expressions

Arithmetic expressions are rules for computing numerical values. They are formed from variables, numbers, functions, and symbols representing addition, subtraction, multiplication, division, and exponentiation. For example, in the expression

$$\text{PREM}^{\wedge}\text{HRS} * 2.50 + \text{OT}^{\wedge}\text{HRS} * 3.75$$

PREM[^]HRS and OT[^]HRS are variables; 2.50 and 3.75 numbers; and + and * symbols for addition and multiplication. If PREM[^]HRS were 40 and OT[^]HRS were 4, the expression becomes 40 * 2.50 + 4 * 3.75 and after performing the arithmetic, reduces to the value 115.00. To save this value, a programmer might write

$$\text{GROSS}^{\wedge}\text{PAY} = \text{PREM}^{\wedge}\text{HRS} * 2.50 + \text{OT}^{\wedge}\text{HRS} * 3.75.$$

The presence of the = symbol tells the compiler to assign 115.00 to the variable GROSS[^]PAY. When expressions are written in this form, they are called "assignment statements".

The arithmetic permitted in an expression is stated by the following symbols:

<u>Symbol</u>	<u>Meaning</u>
+	Addition
-	Subtraction
*	Multiplication
/	Division
**	Exponentiation

In addition to arithmetic, the following mathematical functions may be used.

<u>Symbol</u>	<u>Function</u>
SIN	Sine
COS	Cosine
ATAN	Arctangent
SQRT	Square Root
EXP	Exponential
LOG	Common Logarithm
LN	Natural Logarithm
ABS	Absolute Value

Arithmetic expressions are evaluated from left to right according to the following priority:

1. Exponentiation and Functions
2. Multiplication and Division
3. Addition and Subtraction

Parentheses may be used to establish a precedence other than the one above. When they are used, the evaluation is performed from the innermost to the outermost pair but still from left to right within a given pair.

Relational Expressions

A relational expression is a statement of magnitude between two values. For example, FICA GR 144.00 is a comparison between the variable FICA and 144.00. The symbol GR stands for the relation "greater than". Other relations may be stipulated by

<u>Symbol</u>	<u>Relation</u>
EQ	Equal to
GR	Greater than
LS	Less than
NEQ	Not equal to
NGR	Not greater than
NLS	Not less than

To have meaning, relational expressions must be stated as conditions. The expression FICA GR 144.00 tells us nothing. However, when it is written as

IF FICA GR 144.00, GO TO ADJUST-PAY

we know immediately what is intended. By definition then, relational expressions are conditions and when evaluated always give a truth-value.

Relational expressions may be explicitly stated or implied. FICA GR 144.00 is an explicit statement of magnitude. In the program example of Fig. 2, implied relations were stated by the words FEMALE, PROGRAMER, ANALYST, and MANAGER. An implied expression is formed by giving a name to a value, a range of values, or to a series of values and ranges. Once the name and its values are defined in the data division, it may be used to mean its associated values. Implied relations are termed "condition-names" since a name was given to a condition, i.e., a value, of a variable. The

variable from which the value is taken is called a "conditional variable". Therefore, writing PROGRAMMER (fig.2) in a decision table block is the same as writing an expression which will compare the TITLE field with the value associated with the title, programmer.

Logical Expressions

Logical expressions provide a convenient method for obtaining truth-values. They are formed by combining true-false variables and relational expressions with the logical operators AND, OR, and NOT. The expression (Fig.2)

PROGRAMMER OR ANALYST

is a logical expression which is true when an employee's TITLE field indicates that he is either a programmer or an analyst.

The rules governing the evaluation of logical expressions may be expressed as follows:

p	F	F	T	T
	F	T	F	T

NOT p	T	T	F	F
p AND q	F	F	F	T
p OR q	F	T	T	T

where p and q are a combination of true-false variables, relational expressions, or logical expressions.

Logical expressions are evaluated from left to right with the logical operator AND having precedence over the OR. Parentheses may be used for grouping or establishing a precedence of evaluation other than the one mentioned previously. When they are used, the evaluation proceeds from left to right from the innermost pair to the outermost pair.

IV TABLE ENTRIES

The previous section outlined the elements of the General Compiler language and briefly showed how they might be used. In the introduction, it was mentioned that these same elements may be employed within the blocks of decision tables. The purpose of this section is to show how this may be done.

Formation of Conditions

By definition, a condition is a relation between a primary block entry and some corresponding secondary block entry. A condition, like a relational expression, may be either true or false. True conditions are said to be "satisfied" and false conditions "not satisfied". From this definition, a condition may be either a relational expression, a logical expression, or a true-false variable since these are the only elements that yield a truth-value.

The formats noted below show how these expressions may be split between primary and secondary blocks to form conditions. In these examples, the word "operand" stand for either a variable (data name or subscripted data name), a constant (literal or named constant), or an arithmetic expression. The word "relation" signifies one of the relational operators - EQ, GR, LS, NEQ, NGR, or NLS. Since arithmetic expressions may be operands of relational expressions and relational expressions as operands of logical expressions, it necessarily follows that arithmetic expressions may appear in logical expressions.

Format

Operand-1 Relation
Operand-2

Example

LEVEL EQ
10

Unit

Example

Operand-1
Relation Operand-2

EXPERIENCE
GR 4

Operand-1 Relation
Operand-2 OR Operand-3

TOTAL (I) NLS
PT(1) OR PT(2) or PT(3)

Operand-1
Relation-1 Operand ₂ OR Relation-2 Operand-3 ...

(X+Y) ** 3
GR P+1 OR LS Q(I)

No Entry
Condition-name

PROGRAMMER

NOT
Condition-name

NOT
FEMALE

No Entry
True-False Variable

REQ=1

NOT
True-False Variable

NOT
END INVENTORY FILE

No Entry
Logical Expression

PROGRAMMER OR ANALYST

NOT
Logical Expression

NOT
X GR Y OR X LS (Z+1)

Formation of Actions

Actions are statements of the things to be done when all the conditions of a row are satisfied. The scope of an action may be one of three kinds: implied assignment, procedural, or input-output. The only action presented so far was assignment. The other two are extensions of General Compiler sentences and will be mentioned here only briefly. The compiler manual should be consulted for a more detailed presentation.

1. Value Assignment. Value assignment is an implied function between associated, primary and secondary block entries. By placing a data name in a primary block and some number in a secondary block, for example, I and 1 of Fig. 2, the compiler automatically produces coding to assign the number to the data name. In the case of our example, 1 is assigned to the subscript I. Other examples of value assignment are given below. In these formats the word variable implies either a data name or a subscripted data name and the word constant either a literal or a named constant.

<u>Format</u>	<u>Example</u>				
<table border="1"><tr><td>Variable</td></tr><tr><td>Constant</td></tr></table>	Variable	Constant	<table border="1"><tr><td>I</td></tr><tr><td>1</td></tr></table>	I	1
Variable					
Constant					
I					
1					
<table border="1"><tr><td>Constant</td></tr><tr><td>Variable</td></tr></table>	Constant	Variable	<table border="1"><tr><td>"COPPER"</td></tr><tr><td>MATERIAL</td></tr></table>	"COPPER"	MATERIAL
Constant					
Variable					
"COPPER"					
MATERIAL					
<table border="1"><tr><td>Variable</td></tr><tr><td>Arithmetic Expression</td></tr></table>	Variable	Arithmetic Expression	<table border="1"><tr><td>ALPHA (I,J,K)</td></tr><tr><td>SIN THETA + (X/P)**2</td></tr></table>	ALPHA (I,J,K)	SIN THETA + (X/P)**2
Variable					
Arithmetic Expression					
ALPHA (I,J,K)					
SIN THETA + (X/P)**2					

Format

Example

Arithmetic Expression
Variable

PI * R**2
AREA*1

True-False Variable
Truth-Value 1 or 0

SWITCH*7
1

Truth-Value 1 or 0
True-False Variable

0
BETA*REQ

2. Procedural actions. Procedural actions provide the means for interrupting the normal execution sequence of a table. Any of the following compiler verbs may be used for this purpose.

GO TO
PERFORM
STOP

The GO verb stipulates an unconditional transfer to a specified part of the table or program. Its destination may be a sentence name, table name, or the row number of a particular table. The format of the GO entry is as follows:

Format

Example

GO TO
Sentence Name

GO TO
TYPE/OUT

GO TO
Table Name

GO TO
TABLE 23

GO TO
Row of Table

GO TO
ROW 7 TABLE BETA

The other form of a procedural control is the PERFORM verb. The PERFORM specifies a transfer to some destination, the execution of a table or a set of sentences at that destination, and a return to the action block following the PERFORM. The sentences or tables acted upon are by definition a "closed procedure" - i.e., they have a single entrance point and a defined exit point. Conventions for writing closed procedures are given in the next section. Legitimate forms of the PERFORM action are

<u>Format</u>	<u>Example</u>
PERFORM <hr/> Sentence Name	PERFORM <hr/> GROSS PAY
PERFORM <hr/> Table Name	PERFORM <hr/> ERROR TABLE

The STOP verb may also be used as an action. It may be placed in either a primary or secondary block. When it is used, no other action may appear with it in the same action column. The STOP terminates processing temporarily or permanently according to what action is taken at the computer's console.

3. Input-Output Actions. Input and output actions are compiler verbs that control the flow of data to and from the computer. They read, write, and validate tape labels of data files assigned to peripheral input-output devices. When data files are referred to from an action block, they must be defined according to the environment and data division specifications listed in the General Compiler manual. The formats of input-output actions are illustrated by the following:

<u>Format</u>	<u>Example</u>
READ <hr/> File Name	READ <hr/> MASTER~FILE
OPEN INPUT or OUTPUT <hr/> File Name	OPEN INPUT <hr/> MASTER~FILE
CLOSE <hr/> File Name	CLOSE <hr/> MASTER~FILE
File Name <hr/> READ, CLOSE, or OPEN verbs	MASTER~FILE <hr/> READ
WRITE <hr/> Record Name	WRITE <hr/> DETAIL~LINE
Record Name <hr/> WRITE	TRANSACTION <hr/> WRITE

The Skip and Repeat Operators

The skip operator makes it possible to show that a condition or action is not to take part in the evaluation of a row. This is done by placing a hyphen (~) in the concerned condition or action block. The compiler then will skip this block and proceed to the next.

The repeat operator is a shorthand method to indicate that a condition or action in the block above is repeated. This is shown by entering a ditto mark (") in the block below the one that is to be repeated. This notation was used with the GO TO action in the sample table of Fig. 2.

Up until now, only components of tables were presented. It was learned in Section II that General Compiler sentences could be used to support the conditions and actions of tables, and the preceding section mentioned tables as closed procedures. This section relates these topics to tables and tables to compiler programs.

Block Conventions for Writing Expressions

1. Words, abbreviations, and symbols of the compiler's vocabulary should not be used as names. They may be combined with other characters to form names.

2. The words in an expression should be separated by at least one space. More than one space is permitted. The space separator is optional if the words are bound by

+ - * / ** ~ () " = , |

3. Subscripts should be enclosed in parentheses. They may be written adjacent to (without a space separator) or apart (with space separators) from their associated data names. Individual subscripts in a list of subscripts should be separated by commas.

4. When two arithmetic expressions appear side by side as in a series, they should be separated by commas.

5. All columns of a table should be bound by the vertical table line, | (12-4-8 punch).

6. The skip and repeat symbols, ~ and ", should be the only entry, other than spaces, in a block.

Conventions for Placing a Table in a Program

1. Tables are written on the General Compiler Sentence Form.
2. A table is preceded by the word TABLE. Naming tables is optional. When a table is given a name, the name may precede or follow the word TABLE. The word

TABLE,
name TABLE, or
TABLE name

should be followed by a period.

3. The table's size is given next and should be placed on the same line as the table's name. The size may be written in one of two ways:

kkk CONDITIONS mmm ACTIONS nnn ROWS.

or

(kkk, mmm, nnn).

Both forms are terminated by a period. The order of writing the number of conditions, actions, and rows is optional in the first case since each can be identified. However, order is important in the second form since the compiler interprets the first number enclosed in parentheses as the number of conditions, the second as actions, and the third as rows. Conditions, actions, and rows are numbered sequentially beginning with 1. Row 1 is the first secondary row; the primary row is not counted in the row count.

4. General Compiler sentences should not be placed between the word TABLE and the primary row of the tables.
5. The double vertical lines that separates conditions from actions may be represented by one or two 12-4-5 punches.

6. The size of each block may vary from column to column and row to row.
7. The only limit on the size of a table is row width. Since the compiler prints a listing of compilation, the recommended row width is 120 characters including card sequence number. Maximum row width is 1200 characters.

Since the table form is an image of an 80-column punched card, a hyphen (~) is placed in column 7 of the form to show that a row is contained on more than one card. In this case, no table column may be split across cards. Each card is to contain a sequence number to insure proper card order. When rows exceed one card, the sequence number of the first card is only printed. Sequence numbers of succeeding cards are stripped out. The row is then printed as a multiple of 120 characters with an integral number of table columns per 120 characters.

8. Expressions too long or complex to be written in blocks may be written after the table's name and size and be executed from the table by means of the PERFORM verb. In addition to expressions, any General Compiler sentence may be used and executed in this manner. To indicate the start of the table the word BEGIN is to follow the list of expressions and sentences. This format may be illustrated by the following:

TABLE name. kkk CONDITIONS nnnn ACTIONS mmm ROWS.

... General Compiler Sentences and Expressions - May be executed only from the confines of the table.

BEGIN

DECISION	TABLE

Closed Procedures

Fig. 4 outlines the format of a closed procedure. By definition a closed procedure may be acted on only by the PERFORM verb. It contains one entrance point and one exit point. In fig. 4 these are indicated by the words BEGIN and END TABLE name. BEGIN and END also act as sentence names and may be referred to from within the procedure body.

Expressions too long to be placed in the blocks of a table may be written in the procedure head and executed from the procedure body by means of the PERFORM verb. As such, they must be given names. In addition to expressions any General Compiler sentence may be written in the head and executed accordingly.

The procedure body contains the table. As shown in Fig. 4 compiler sentences may precede and follow the table. Execution is sequential starting with the sentence or table after the word BEGIN and proceeds until the exit END TABLE is reached. It is at this point that control is reverted to the PERFORM verb which originally referenced the procedure. Any unconditional transfer from within the procedure to the outside is undefined. However, PERFORM verbs in the body may reference other closed procedures.

Closed procedures should be written apart from the main program.

DECISION TABLE AS A CLOSED PROCEDURE

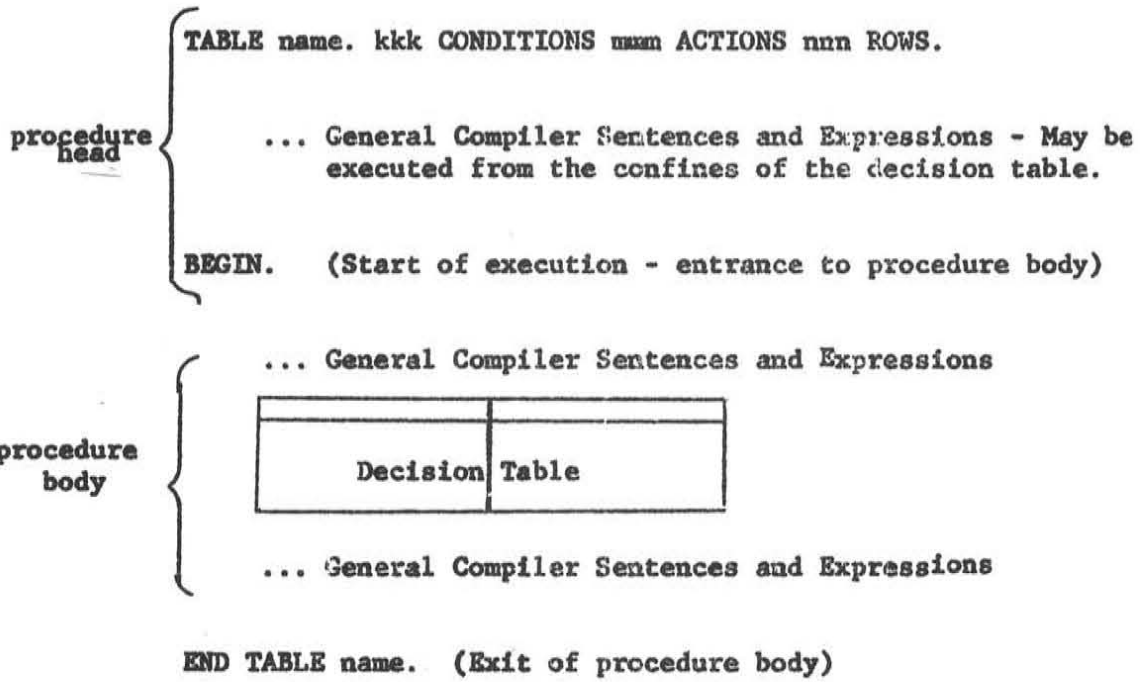


Fig. 4

IBM Data Processing Division
Thomas J. Watson Research Center
P. O. Box 218
Yorktown Heights, New York

June 23, 1961

Memorandum to:

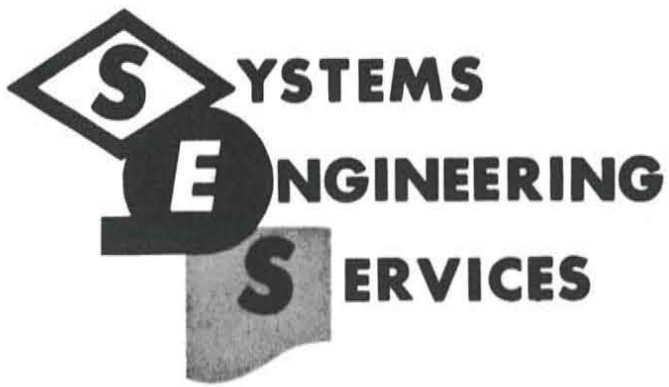
Subject: Tabular Techniques Development
 Distribution #3

This is the third release of material concerning the development of tabular techniques for systems and programming description. Enclosed are three reports:

- (1) A report by Sutherland Company describing a method of recording management decision rules and other information necessary to adapt an information system to an automatic medium of data processing.
- (2) A report by Burton Grad, IBM, describing two techniques of representing the decision logic of an insurance company file maintenance problem; namely, traditional flow charts and tabular form.
- (3) A paper given by Burton Grad at the 12th GUIDE International meeting in Montreal, Canada, June 1, 1961 describing the general concept of tabular techniques.



Burton Grad, Manager
Systems Engineering Services



CLEARINGHOUSE REPORT

INFORMATION PROCESSING SYSTEM
ANALYSIS

June 5, 1961
Ref. No. 1F2

Sutherland Co.

INTERNATIONAL BUSINESS MACHINES CORPORATION
White Plains, New York

This material is distributed to keep IBM personnel informed of new developments. Selection is based on interest; this department makes no claim for the desirability of this approach nor necessarily recommends its use.

If additional copies are desired, please contact the Clearinghouse. No part of this material should be reproduced or distributed outside IBM without approval of the Clearinghouse.

INFORMATION PROCESSING SYSTEM ANALYSIS INSTRUCTIONS

1. Purpose. To provide a standard method of recording the management rules (arithmetic and decision processes) and other information necessary to adapt an Information System to a mechanical or other medium of processing.

2. General. The method described in the following instructions eliminates the need for lengthy narrative with its inherent disadvantages of misinterpretation by the reader and difficulty of organization by the writer. This method also eliminates the need for the system analyst to prepare detailed flow charts to convey to a processing specialist the processing required to obtain the desired results of the Information Processing System. The method of documentation is general enough to allow the Information System to be adapted to any medium of processing, but detailed enough to permit the application of the Information System to electronic machine processing by a machine specialist who has no prior knowledge of the Information System.

A. Documentation Preparation. The documentation will be prepared by the system analyst and forwarded to the processing specialist. The processing specialist may be a punched card equipment specialist, an electronic equipment processing specialist or a manual and standard office equipment processing specialist. In many instances, the manual and standard office equipment processing specialist will be the system analyst.

B. Content of Documentation. The documentation prepared by the system analyst will include the following:

(1) General System Chart including the inputs to the system and the sources of the inputs, the outputs of the system and the disposition of the outputs and the data to be retained by the system.

(2) A general narrative description of the Information System which will include the purpose and scope of the Information System and any other pertinent information that may be helpful to the Processing Specialist.

(3) Description Sheets

- a. Input Description
- b. Process Description and Process Description Continuation
- c. Output Description

(4) Any reference notes that are required to clarify the Input, Output or Process Description sheets.

(5) A sample copy of each "hard copy" input and "hard copy" output of the Information System. Element codes will be entered on the input and output samples to identify the elements and their position.

Note: Appendix II is a sample of the documentation for an Information Processing System.

3. Input Description Sheet.

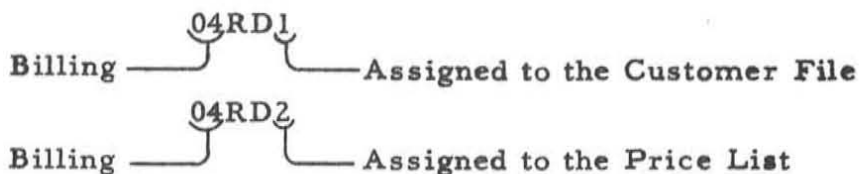
A. General. An Input Description Sheet is used to describe the content of Action Sets and Retained Data Sets which are input to the information system.

B. Headings.

(1) In the upper left-hand corner, place the two-character "System Identification" for the system being described.

(2) Below the "System Identification", place the "Set Identification" for the Input Set being described. If the input is an Action Set, use the identification of the Action Set. If the input is a Retained Data Set, use the unique Retained Data Set identification assigned to the set.

The first two characters of the Retained Data Set identification are the System Code, the next two characters will be "RD". The next character (s) is used to identify uniquely each Retained Data Set. For example:



(3) Indicate in the space provided for "Frequency of Processing" the most frequent period in which this set is to be input to the system.

(4) Process. Indicate in the space provided the name of the process being documented. In most instances the process will directly correspond to what is described by the System Identification. Occasionally the System Identification is not definitive of the process being documented and the actual process name should be indicated. For example:

System Code 20 is assigned to Salary Payroll which includes: Pay Check Preparation, Personnel Reports, Labor Distribution, Tax Reports and Annuity Reports. In this example, the System Code would be 20, but the process would be Pay Check Preparation, Labor Distribution, etc. depending on the process being documented.

(5) Place the "Set Name" in the space provided.

(6) Indicate in the space for "Volume" the "Average" and "Peak" number of sets that will be available as input in the time period shown for "Frequency of Processing".

(7) In the space provided indicate the Form Type for the set. Examples of "Form Type" are: "Manual", "Punched Card", "Magnetic Tape", and "Paper Tape".

(8) For "Source System I. D." indicate the two-character System Code of the System which processes the set immediately prior to this system. If the Input Set is a Retained Data Set which is added to in more than one system, indicate the system from which the Retained Data Set will be received.

(9) In the upper right-hand corner indicate the page number, the name of the person preparing the Input Description Sheet, and the date of preparation.

C. Management Rule Numbers. For Action Sets, ^{AND RETAINED DATA SETS} indicate in the spaces provided across the top of the sheet the three-digit numbers of the Management Rules (other than Validation Rules) which must be executed if this set is present. If there is not sufficient space on one Input Description Sheet for all the rules, use additional sheets.

D. Element Name.

(1) In this column enter the "Element Names" assigned to the elements that are contained in the Input Set. For an input, regardless of whether or not space is provided for an element, no entry should be made for the element, if it is always blank.

(2) Additional information on each element is placed to the right of the element name.

E. Element Code. In this column place the seven-character element code number corresponding to each element name.

F. Element Code - Suffix.

(1) An element in a set may be used differently or prepared differently depending on what other elements identify it. An example is the Element "Quantity on Hand Total". This element may appear twice on a set. In one place, it may be the total "Quantities on Hand" for each "Stock Number" at each "Location". In the other place, it may be the total of all "Quantities on Hand" for each "Stock Number" at all "Locations". In the first instance, location would be an Identifying Element; in the second, it would not. To indicate this difference for the element in this set, two suffixes "A" and "B" would be assigned. For each different grouping of Identifying Elements for an element, assign a different suffix, beginning with "A". (See paragraph 3, N, (3) following).

G. Element Description - Alpha. If the element described by the element name contains any non-numeric characters, enter an "A" in this column. Otherwise, leave the column blank.

H. Element Description - Numeric. If the element described by the element name contains any numeric characters, enter an "N" in this column. Otherwise, leave the column blank.

I. Element Description - Characters - Total. Place in this column a maximum of two digits to describe the maximum number of characters that the element may contain. Do not include in the total number of characters, punctuation marks in numeric fields which are used for arithmetic processes.

J. Element Description - Characters - Decimal. This entry is made only for all numeric elements which may be used in arithmetic computation. Enter in this column the number of digits that appear to the right of the implied decimal point.

K. Element Classification (Class.). Depending on whether the element described by the element name is a Recognition, Identification, Action, Action Modifier, Information, or Superfluous Element, enter an "X" in the appropriate column. See the definitions for the different Element Classifications in Appendix I. Generally, the different classifications are mutually exclusive. However, any element may be described by more than one classification other than "Information" and "Superfluous". For retained Data Sets only Recognition and Identifying Elements need be indicated.

L. Number of Times an Entry May Appear on This Set. Place in this column a maximum of three characters to indicate the "Average" and a maximum of three characters to indicate the "Peak" number of times an entry may appear for this element on this set. If the number exceeds 999, use "C" for hundreds and "M" for thousands.

M. Validation Rule (s). In this column list the three-digit Rule Numbers for the Management Rules which must be executed to validate the element described by the element name. Use as many lines as are necessary for each element name.

N. Identifying Element Codes.

(1) For Identifying Elements that are used to identify an element on the Input Set, the Identifying Element Codes are listed vertically in the spaces provided. If more space is required, use additional Input Description Sheets.

(2) Place an "X" in the "Identifying Element Code" column and Element row intersection if the Identifying Element is used to identify the element indicated on that row. Each entry for the element described by the element name is identified by one combination of entries for the elements described by the Identifying Element Codes.

(3) The first two lines of Figure I illustrates the example described in paragraph 3, F, preceding. Quantity on Hand with Suffix "A" is for each Stock Number at each location. Consequently an "X" appears under both 9300100 and 7976050, the Element Codes for Stock Number and Location respectively. Quantity on Hand with Suffix "B" is for each stock number at all locations. An "X" only appears under 9300100, the Element Code for Stock Number. In this case, Stock Number alone is the Identifying Element for Quantity on Hand. The third line of Figure 1 indicates that the entry (s) for location is identified by an entry for Stock Number.

System Identification _____ Process _____
 Set Identification _____ Set Name _____
 Frequency of Processing _____ Volume: Average _____

Form
Source System I. D.

MANAGEMENT RULE NOS.		ELEMENT DESCRIPTION				IDENTIFYING ELEMENT CO			
ELEMENT NAME	ELEMENT CODE	Char			9300100	7976050			
		Suffix	Alpha	Numeric					
QTY ON HAND TOT	8768150	A							
	8768150	B			X	X			
LOCATION	7976050				X				
STOCK NO	9300100								

FIGURE 1. USE OF ELEMENT SUFFIXES AND IDENTIFYING ELEMENT CODES

O. Reference Note (Ref. Note). If there is a need for a reference note, place a check mark (✓) in the column. Cross-reference the note with the System Identification, Process, Set Identification, and if required the Element Code and Suffix.

P. Remarks. This column may be used for any additional information believed necessary by the analyst preparing the Input Description Sheet.

4. Process Description Sheet.

A. General.

(1) A Process Description Sheet is used to describe Management Rules used in processing information within a system.

(2) Rules for Validation are shown on separate sheets from all other processing rules. It is assumed by the analyst that all Validation processing is to be accomplished before other processing is begun.

B. Headings.

(1) In the upper left-hand corner place the two digit "System Identification" for the Analysis System.

(2) In the space provided for "Process", indicate the name assigned to the process being described. (Refer to paragraph 3, B, (4) preceding).

(3) If the processes described by the Management Rules on the sheet are Validation Processes, place an "X" in the "Validation" Box.

(4) In the right-hand part of the heading, enter in the spaces provided: the page number, the name of the person preparing the sheet, and the date of preparation.

C. Line Number. On each line in this column, place a four-character line number. It is suggested that the right-most digit always be blank in case there is a later need for insertion of additional lines. Line numbers will be uniquely assigned to all lines within the description of a particular ^{MANAGEMENT RULE} process for a system. Thus, if the last line on Page 1 for a process is line number 022, the first line number on Page 2 will be 023.

Examples of line numbers are :

011
012
0131
0132
014

D. Condition/Action Indicator (C/A).

(1) If a condition is expressed on this line, place a "C" in this column; if the line is used to express an action, place an "A" in this column. If what has been placed in this column for an immediately previous line is true for a line that follows, no entry need be made for the line that follows.

E. Management Rule - Current. In this column on the first line for each Management Rule place a three-digit number for the Management Rule. The numbers of all Management Rules will be uniquely assigned for all rules within a Process for a System.

F. Management Rule - Prior. In this column list the three-digit numbers of all of the Management Rules which must be considered before the rule specified in the "Management Rule - Current" column is considered. Generally, a rule is prior to another rule only if it specifies the creation of elements of data necessary for the processing of the current rule. Management Rules for Validation of elements will not be shown as prior rules for non-validation Management Rules.

G. Source - Element Name, Prior Result or Actual Value.

(1) If one source for a condition or action is an element, place the name assigned to the element in this column. If the source is an actual value (Literal or Descriptive constants - See Appendix I) place the actual value in this column. If the source is the result of an action in any rule, place the designation of the result in this column. (Results of an action are designated as "Result X", where "X" is any character A to Z or 0 - 9. The first result of a rule is designated as "Result A", the second as "Result B", etc. Unique designations of prior results are only necessary within each rule. Two different rules may each have an intermediate result designated as Result A.

(2) Deletion of an Element. The deletion of an element from a set is indicated by placing the Descriptive Literal "/BLANK/" in this column, entering a check mark in the "Set Equal To" column, and entering the Element Name and Set Identification for the element to be deleted in the appropriate spaces in the "Source/Disposition" column.

(3) Deletion of a Set. The deletion of a set is indicated by placing the Descriptive Literal "/BLANK/" in this column, entering a check mark in the "Set Equal To" column, and entering the Set Identification for the set to be deleted in the "Source/Disposition - Set Identification" column. In this case the "Source/Disposition - Element Name" is left blank.

H. Source - Element Suffix. If the entry made in the "Source - Element Name Prior Result Actual Value" column was an Element Name, and if a suffix was assigned to the element on the Input Description Sheet, the suffix

which was assigned is entered in this column. Otherwise, the column is left blank.

I. Source - Set Identification.

(1) If the entry made in the "Source - Element Name, Prior Result or Actual Value" column was an Element Name, enter in this column the seven-character set designation for the set of which the element is a part. If an element for a rule may appear in one Input Set or another, depending on which set is present, more than one Set Identification may be listed in this column as a source for the element described by the Element Name. If the entry in the "Element Name" column is the designation of a Result of an action in this rule or another rule, enter the three-digit number for the source rule within parentheses. This column is left blank if the entry made in the "Source" column is an entry for an actual value. Examples of entries that may be made in this column are:

24165A = Set
 (152) = Management Rule
 152 = Set

(2) The addition or insertion of a set into an Output Set or Retained Data Set may be indicated by placing the Set Identification of the set to be added or inserted in the "Source Set Identification" column, the Set Identification of the Output Set or Retained Data Set in the "Source/Disposition Set Identification" column and a check mark in the "Set Equal To" column. The element columns of both the Source and Source Disposition will be left blank. This procedure will only be used if all the Elements of the Output Set or Retained Data Set are contained in the Input Set.

J. Condition (Cond.). If a condition is expressed on a line, it is "Greater Than", "Less Than", or "Equal To". Place a check mark (✓) in the appropriate column (s) to indicate the relationship between the first and the second Source Elements or Actual Values. The relationship between the three conditions is a logical "or" condition. More than one column may be checked for a line. In reading, "or" is inserted between each condition checked.

For example, if "AMT SALARY" is the first Source Element, (O) is specified as the second Source Element (Actual Value), and the "Less Than" and "Equal To" conditions are checked, this is read, "If AMT SALARY is Less Than or Equal To O..."

K. Operation.

(1) To express an Arithmetic Operation for an action relating two elements, results or actual values, place one of the following operation

symbols in the column:

+ for addition
 - for subtraction
 x for multiplication
 / for division
 Σ for sum

(2) Explanation of Operation Symbols.

a. An entry of "+" in this column indicates that the first source entry is to be added to the second source entry.

b. An entry of "-" in this column indicates that the second source entry is to be subtracted from the first source entry.

c. An entry of "x" in this column indicates that the second source entry is to be multiplied by the first.

d. An entry of "/" in this column indicates that the first source entry is to be divided by the second.

e. An entry of " Σ " (Greek letter "Sigma") in this column indicates that all entries for the first specified element are to be summed.

L. Set Equal To. If the element or result specified in the "Source/Disposition" column is to be "Set Equal To" another element, actual value or prior result, or is to be "Set Equal To" the result of an arithmetic action, place a check mark in this column. The last line of any action within a rule will have a check mark in the "Set Equal To" column.

M. Source/Disposition - Element - Name Result, Prior Result or Actual Value.

(1) If the entry to be made in this column is for a source for a condition or an action, the way to make the entry is described in paragraph 4, G, (1).

(2) If this column is used to indicate disposition for a result of an action, enter the appropriate element name or prior result designation. (See "Result X", paragraph 4, G, preceding).

N. Source/Disposition - Element - Suffix. If the entry made in the "Source/Disposition - Element Name" column is an Element Name, and if, on the Input or Output Description Sheet the element has been assigned a suffix, enter the appropriate suffix in this column. Otherwise, leave the column blank.

O. Source/Disposition - Set Identification. If the entry made for the "Source/Disposition" column is an entry for a Source, see paragraph I. If the entry is a Disposition entry for an element, enter in the "Set Identification" column the Set Identification for the set or sets in which the Element is to be placed. If the entry is a Disposition entry for an intermediate Result, leave the "Set Identification" column blank.

P. Operation.

(1) To relate arithmetically an entry in the "Source/Disposition" column on one line with an entry in the "Source" column on the next line, indicate the arithmetic operation in this "Operation" column using one of the following symbols:

- + for addition
- for subtraction
- / for division
- x for multiplication

(2) Explanation of Operation Symbols.

a. An entry of "+" in this column indicates that the "Source" entry on the next line is to be added to the "Source/Disposition" entry on the line where the "+" appears.

b. An entry of "-" in this column indicates that the "Source" entry on the next line is to be subtracted from the "Source/Disposition" entry on the line where the "-" appears.

c. An entry of "/" in this column indicates that the "Source/Disposition" entry on the same line is to be divided by the "Source" entry on the next line.

d. An entry of "x" in this column indicates that the "Source/Disposition" entry on the same line is to be multiplied by the "Source" entry on the next line.

Q. Note Reference (Note Ref. (✓)). If a note or remarks are necessary and/or advisable to explain further a condition or an action, place a check mark in this column. On the sheet where it appears, cross reference the note to the System Identification, Process, ^{MANAGEMENT RULE} and first Line Number of the Condition or Action to which the note applies.

R. Management Rule Suffix and Frequency.

(1) Eighteen Management Rule Suffixes, "A" through "R", are pre-printed across the top of the Process Description Sheet. If more than eighteen

suffixes are necessary for a rule, Process Description Continuation Sheets should be used.

(2) In describing a Management Rule, all the conditions which must be considered at any one time will be listed on a Process Description Sheet. Following the conditions, all the actions which may be executed for the conditions of the rule will be listed on the same Process Description Sheets (insofar as possible). Management Rule Suffixes are used to relate a combination of positive and/or negative results for one or more conditions to the execution of one or more actions within a rule.

(3) Unless a Management Rule describes an unconditional action (action taken regardless of the results of any conditions), an action is taken only when the results of certain conditions are positive ("Y") and/or negative ("N"). In describing a Management Rule, all the pertinent possible combinations of condition results must be related to the actions for the rule.

(4) A simple example is shown in Figure 2. In this sample Management Rule there are only three conditions shown on lines 001 to 003. One set of results for the conditions are listed under Suffix A; i. e., if the result of the conditions on lines 001 and 002 are positive the action specified on line 004 should be taken. Under Suffix C, if the results of the conditions on lines 001 and 003 are positive and the result of that on line 002 is negative, the action specified on line 004 should be taken.

(5) As is evident from the example, pertinent results for conditions are indicated for a suffix using "Y" for "Yes" and "N" for "No". Under each suffix an indication of an action to be taken is shown with an "X" on the line, ("Set Equal" line if more than one) on which the action is described. If neither "Y" nor "N" is placed on the line for a Condition under a given Suffix, it indicates that for the combination of results shown under the suffix, the result of this condition is immaterial; the result can be positive, negative, or undetermined.

(6) For a Management Rule, on the line (s) following the last line describing the conditions, the analyst will indicate the probable Frequency of Occurrence as a percentage for the results of the conditions listed under each suffix. The total Frequencies of Occurrence for all suffixes within a Rule should be 100 percent. For any frequencies less than 1%, use "1". In Figure 2 the Frequency of Occurrence is indicated between lines 003 and 004. In this example the probability of the conditions of rule 001A prevailing is 80%, written " $\frac{8}{10}$ ". For rule 001D, the probability of occurrence is 4%, written " $\frac{4}{10}$ ".

5. Process Description Continuation Sheet.

A. General. A Process Description Continuation Sheet is used only if, for a Management Rule, there were insufficient suffixes on the Process Description Sheet to depict all the combinations of results for the conditions described on it.

B. Headings. The instructions for completing the heading information are the same as shown for the Process Description Sheet, paragraph 4, B, preceding.

C. Line Number. In the line number column post the line numbers from the Process Description Sheet that this sheet is a continuation of. Use exactly the same spacing and relative positioning of the line numbers as appears on the Process Description Sheet. This will enable the user to lay a completed Continuation Sheet next to the sheet it is a continuation of to have effectively a single sheet of paper.

D. Management Rule Suffix and Frequency.

(1) In the blank heading blocks, place one or two-character suffix designations that will be unique for the Management Rule to which they apply. If a two-character suffix designation is used, place the more significant character over the less significant character.

(2) All other information is placed on the sheet as described under Process Description Sheets, paragraph 4, R, preceding.

6. Output Description Sheet.

A. General. An Output Description Sheet is used to describe the content of output from an Information System.

B. Headings.

(1) Enter the "System Identification" for the system being described in the space provided.

(2) Enter the "Set Identification" for the Output Set in the space provided.

(3) Indicate in the space provided the name of the process being documented. (Refer to paragraph 3, B, (4) preceding).

(4) In the space provided for "Number Copies" indicate the number of copies that are required for this Output Set.

(5) Place the "Set Name" in the space provided.

(6) Indicate in the space provided for "Volume" the "Average" and "Peak" number of sets that will be prepared.

(7) Form Type. Indicate the Form Type for the set. For example, Standard Print, Punched Card, Multilith Mat, etc.

(8) Special Form I. D. If the set is to be prepared on a special form, indicate the identification of the special form in the space provided.

(9) In the upper right-hand corner enter the Page number, the name of the person preparing the sheet, and the date of preparation.

C. Element Name.

(1) In this column enter the Element Name for each of the elements which may appear in this set.

(2) Additional information on each element is placed to the right of the Element Name.

D. Element Code.

(1) In this column enter the seven-character Element Code Number corresponding to each Element Name.

E. Element Code - Suffix.

(1) If an Element Code Suffix is required (See paragraph 3, F, Input Description Sheet), enter a one-character alphabetic designation for the suffix in this column.

F. Element Description - Alpha.

(1) If the Element described by the Element Name contains any non-numeric characters, enter an "A" in this column. Otherwise, leave the column blank.

G. Element Description - Numeric.

(1) If the Element described by the Element Name contains any numeric characters, enter an "N" in this column. Otherwise, leave the column blank.

H. Characters - Total.

(1) Enter in this column a maximum of two digits to describe the maximum number of characters that the Element may contain.

I. Characters - Decimal.

(1) An entry is made in this column only for all numeric Elements which are a result of or may be used in arithmetic computations. Enter in this column the number of digits that should appear to the right of the implied decimal point.

J. Element Classification.

(1) Depending on whether the Element described by the Element Name is a "Recognition", "Identification", or "Other" classification of Element, enter an "X" in the appropriate column.

K. Number of Times an Entry May Appear on This Set.

(1) Enter in this column a maximum number of three characters to describe the "Average" and a maximum of three characters to describe the "Peak" number of times an entry may appear in the Set for the element described by the Element Name. If the number for either exceeds 999, use "C" for "hundreds" and "M" for "thousands".

L. Source - Set Type.

(1) If the Source for the element described by the Element Name is other than "Direct Recording" from an Action Set or Retained Data Set, place an "X" in the column headed "Process (X)".

(2) If the element described by the Element Name is to be placed on the Output Set as a result of a Direct Recording from a Retained Data Set after all posting to the Retained Data Set has been accomplished, enter an "A" in the column headed "Ret'd (A, B, or X)".

(3) If the element described by the Element Name is to be placed on the Output Set as a result of a Direct Recording from a Retained Data Set before any posting to the Retained Data Set has been accomplished, enter a "B" in the column with the heading "Ret'd (A, B, or X)".

(4) If the element described by the Element Name may be placed on the Output Set as a result of a Direct Recording from a Retained Data Set, either before or after posting to the Retained Data Set has been accomplished, enter an "X" in the column headed "Ret'd (A, B, or X)".

(5) If the element described by the Element Name is placed on the Output Set as a result of a Direct Recording from an Action Set, enter an "X" in the column headed "Action (X)".

M. Source - Source Set Identification for Direct Recording.

(1) If the element described by the Element Name is to be placed on the Output Set as a result of Direct Recording from a Retained Data Set or an Action Set, enter in this column a maximum of seven characters for the Set Identification of each source set. If there are more than three sources, use additional lines.

N. Identifying Element Codes.

(1) For Identifying Elements that are used to identify Elements on the Output Set, the Identifying Element Codes are listed vertically in the spaces provided. If more space is required, use additional Output Description sheets.

(2) Place an "X" in the Identifying Element Code column and Element row intersection if the Identifying Element is used to identify the Element indicated on that row.

O. Reference Note.

(1) If there is a need for a "Reference Note", place a check mark in this column. Cross-reference the note using the System Identification, Process, Set Identification and if necessary, the Element Code and Suffix.

APPENDIX I - DEFINITIONS

1. Action Element

An element within an Action Set, the entry for which is the value to be inserted or replaced, or the value of the adjustment to be made via a Recording Action or Actions or arithmetic computation.

2. Action Modifier Element

An element within an Action Set which alters the Recording Action or Actions in some manner.

3. Action Set

An Input Set for a system whose presence may require the execution of specific Management Rules. Input other than Retained Data Set.

4. Constant Value

A value, which does not appear as an element in either a Retained Data or Action Set, used as a source for an element or elements in an Output Set.

A. Descriptive Constant

An entry which designates between two slashes (/) the commonly understood name of a constant value.

Examples are :

- / Blank / - Designates one or more blanks.
- / Current Year / - Designates 1962, if that is the current year.
- / ANNN / - Designates a field in which the first character is non-numeric and the rest are numeric.

B. Literal Constant

The designation of a constant value between parentheses where the constant value is identical to what appears between the parentheses.

An example is :

(06) which designates a constant value of "06".

5. Direct Recording

The unconditional transferring of an element from an Action Set or Retained Data Set to an Output Set. No prior processing other than validation is required for the element in the Action Set or Retained Data Set. The recording is dependent on the presence of the Action Set or Retained Data Set and the requirement to produce the Output Set.

6. Frequency of Occurrence

A number which indicates, as a percentage, the probability a particular result, or combination of results of a condition or conditions, will prevail.

7. Identification Element

An element within an Action Set which permits the segregation of a particular set from others containing the same Recognition Element values; it is used to associate the set with other sets containing different Recognition Element values and to indicate how elements within the set are recorded and identified.

8. Information Element

An element within an Action Set, which does not influence the Recording Action nor is it recorded in this system. It may be subject to validation for the purpose of an overall system check and is required for processing in subsequent systems.

9. Management Rule

The action or actions and generally an associated condition or conditions which indicate the decisions and processes required to operate an Information Processing System.

10. Output Set

A set created by an Information Processing System for the use of another Information Processing System or by the same Information Processing System, but using a different medium to accomplish its processes.

11. Process

The production of elements of data through the execution of Management Rules. Includes all data processing except Direct Recording.

12. Recognition Element

An element within an Action Set which identifies the function of the set. The Set Identification is a Recognition Element unless otherwise stated.

13. Retained Data Set

A set which is used to maintain elements which are required to accomplish the preparation of the Output Sets and may not be available on the Action Sets. The Retained Data Set will include the elements required to validate the Action Sets.

14. Set

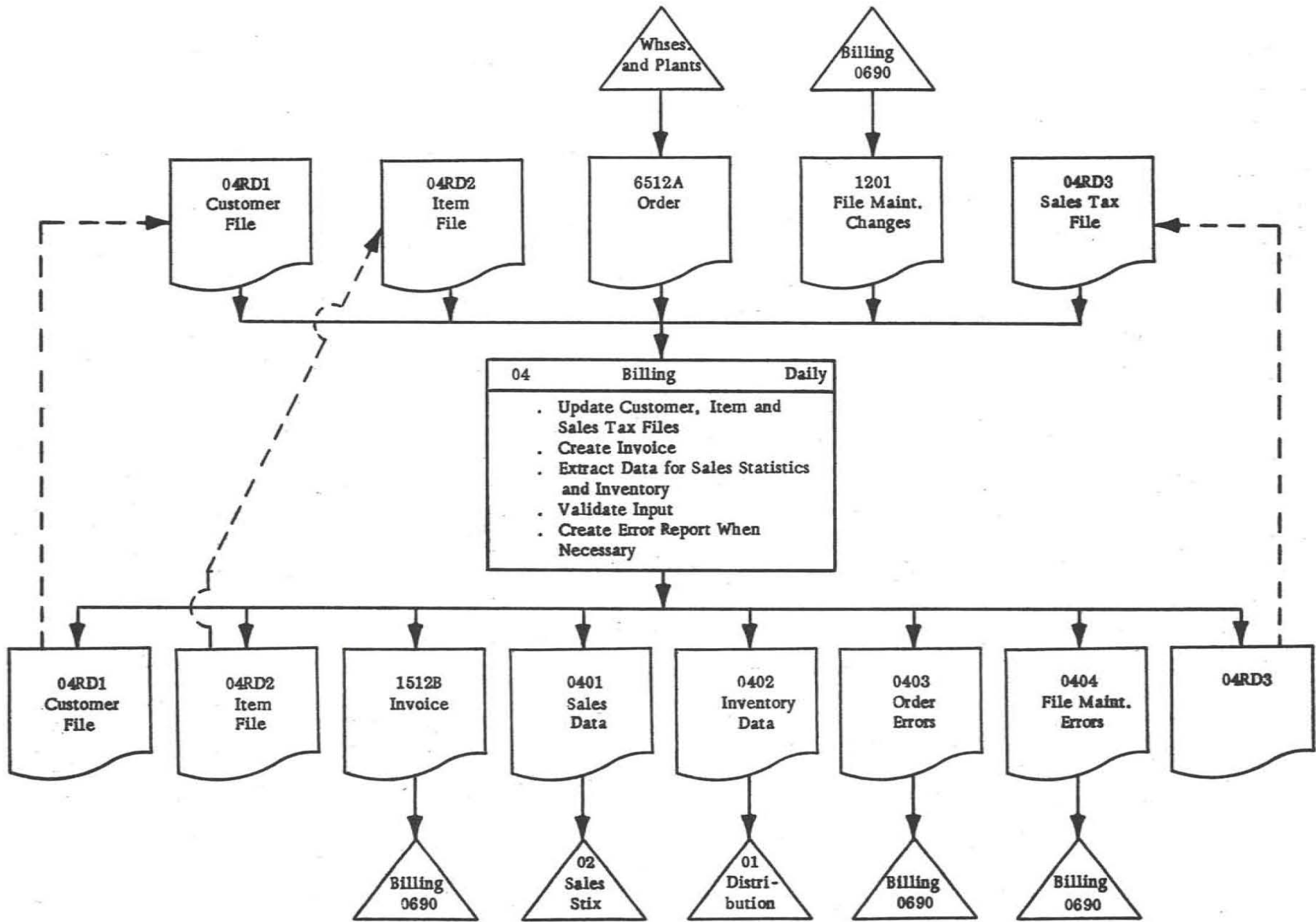
A meaningful grouping of more than one element of data.

15. Superfluous Element

An element within an Action Set which is not required for processing in this or subsequent systems.

SYSTEM CODE 04 BILLING

1. Purpose: To develop an invoice from a copy of the order which indicates that shipment has been made to a customer from a warehouse or factory.
2. Scope: The system will include all debit billing to all customers.
3. Other Outputs:
 - a. Selected data will be furnished to the Sales Statistics system for Sales Accounting and Sales History.
 - b. Selected data relative to inventory will be furnished to the Distribution system for inventory adjustments.
 - c. A record of input received that did not meet the criteria established (invalid) will be furnished to the Billing Department.



Form 1201

Customer Code	Disc. %	Change Code
<input type="text"/>	<input type="text"/> ← 6920130	<input type="text"/>
↑ 8981100 6813250 ↑		8760100 ↗
Name Sold To		
<input type="text"/>		
<input type="text"/>		
Address Sold To		
<input type="text"/>		
<input type="text"/>		

If Customer Code, Name Sold To or Address Sold To changes, create

1. "Delete" for Old Customer Code
 2. "Add" for New Customer Code, Disc. %, Name Sold To and Address Sold To
- Change Codes: 1 = Delete 2 = Add 3 = Change Disc. %

6404

CUSTOMER CHANGE ERRORS

8981100

6813250

8760100

7000100

6920130

6813200

Current
Month Day Year
↓ ↓ ↓
xx - xx - xx

xxxxx

xxxx bb

X

bb xxxxxxxx bb xx.x bb

xx - - - xx

xx - - - xx

6813210

System Identification 0A Process BILLING INPUT DESCRIPTION

Page 1 of 23

Set Identification 1512A et Name ORDER

Form Type MANUAL

Prepared by: HCD

Frequency of Processing DAILY Volume: Average 500 Peak 1000

Source System I. D. 01

Date 6/15/60

MANAGEMENT RULE NOS.		006	007	008	009	010	011	012														
ELEMENT NAME	ELEMENT CODE	ELEMENT DESCRIPTION		ELEMENT CLASS. (X)		NO. TIMES AN ENTRY MAY APPEAR ON THIS SET		Validation Rule (S)	IDENTIFYING ELEMENT CODES (X)										REMARKS			
		Suffix	Char.	Recog.	Action Mod.	Average	Peak		8368100	9900100										Ref. Note (✓)		
		Alpha	Total	Ident.	Info.																	
ACCT CENTER	6017100	A	13				X	1	1													
MO CUST ORD	8168620	N	02 0				X	1	1													
DA CUST ORD	6836520	N	02 0				X	1	1													
YR CUST ORD	9980620	N	02 0				X	1	1													
ORDER NO	8368100	AN	09			X		1	1													
MO SHIPPED	8168650	N	02 0				X	1	1	001												
DA SHIPPED	6836550	N	02 0				X	1	1	001												
YR SHIPPED	9980650	N	02 0				X	1	1	001												
INVOICE NO	7680100	AN	09				X	1	1													
CUST NAME SOLD	6813200	A	70				X	1	1													
CUST ADDR SOLD	6813210	AN	70				X	1	1													
CUST NAME SHIP	6813220	A	70				X	1	1													
CUST ADDR SHIP	6813230	AN	70				X	1	1													
CUST ORDER NO	6813100	AN	12				X	1	1													
SALESMAN NO	8981100	N	05 0		X			1	1	003												
CUST ACCT NO	6813250	N	04 0		X			1	1	003												
TERMS PAYMENT	9496100	AN	23				X	1	1													
SHIPG TERMS	9111100	A	23				X	1	1													
CARRIER NAME	6466100	A	24				X	1	1													
SHIP FROM TOWN	9110120	A	13				X	1	1													
SALES TAX CODE	9464160	AN	05		X			1	1	002												
PRICE BASE	8664100	AN	08				X	1	1													
SIZE	9144100	AN	21				X	3	15													
LINE NAME	7941100	AN	25				X	3	15													

System Identification 04 Process BILLING INPUT DESCRIPTION
 Set Identification 1512 A st Name ORDER
 Frequency of Processing DAILY Volume: Average 500 Peak 1000

Form Type MANUAL Page 2 of 23
 Source System I. D. 01 Prepared by: H C J
 Date 6/15/60

MANAGEMENT RULE NOS.		006	007	008	009	010	011	012																	
ELEMENT NAME	ELEMENT CODE	ELEMENT DESCRIPTION			ELEMENT CLASS. (X)				NO. TIMES AN ENTRY MAY APPEAR ON THIS SET		VALIDATION RULE (S)	IDENTIFYING ELEMENT CODES (X)								REMARKS					
		Suffix	Alpha	Numeric	Char.	Decimal	Recog. Ident.	Action Mod.	Info.	Superfluous		Average	Peak	8368100	9300100									Ref. Note (✓)	
STOCK NO	9300100	A	N	07			X			3	15	004	X												
QTY ORDERED	8768500		N	05	0		X			3	15	005	X	X											
QTY SHIPPED	8768550		N	05	0		X			3	15	005	X	X											
DISC PC QSA	6920110		N	02	1		X			1	1		X												

Appendix II, Page 6

System Identification 04 Process BILLING INPUT DESCRIPTION

Set Identification 04RDI Set Name CUSTOMER FILE

Frequency of Processing DAILY Volume: Average 20,000 Peak 30,000

Form Type MAG T.

Page 3 of 23

Source System I. D. 04

Prepared by: HCU
Date 6/15/60

ELEMENT NAME	ELEMENT CODE	ELEMENT DESCRIPTION				ELEMENT CLASS. (X)				NO. TIMES AN ENTRY MAY APPEAR ON THIS SET		VALIDATION RULE (S)	IDENTIFYING ELEMENT CODES (X)						Ref. Note (v)	REMARKS							
		Suffix	Alpha	Numeric	Char.		Recog.	Ident.	Action	Action Mod.	Info.		Superfluous	Average	Peak	8981100	6813250										
					Total	Decimal																					
SALESMAN NO	8981100			N	05	0	X					1	1												✓		
CUST ACCT NO	6813250			N	04	0	X					1	1	X													
CUST NAME SOLD	6813200			A	70							1	1	X	X												
CUST ADDR SOLD	6813210			AN	70							1	1	X	X												
DISC PC SOP BON	6920130			N	02	1						1	1	X	X												

System Identification 04

Process BILLING

INPUT DESCRIPTION

Page 4 of 23

Set Identification 04RD-2

Name ITEM FILE

Form Type MAGT.

Prepared by: HCL

Frequency of Processing DAILY

Volume: Average 3000 Peak 4500

Source System I. D. 04

Date 6/15/60

MANAGEMENT RULE NOS.

ELEMENT NAME	ELEMENT CODE	ELEMENT DESCRIPTION					ELEMENT CLASS. (X)					NO. TIMES AN ENTRY MAY APPEAR ON THIS SET		VALIDATION RULE (S)	IDENTIFYING ELEMENT CODES (X)										Ref. Note (✓)	REMARKS		
		Suffix	Alpha	Numeric	Char.		Recog.	Ident.	Action	Action Mod.	Info.	Superfluous	Average		Peak	9300100												
					Total	Decimal																						
STOCK NUMBER	9300100	A	N	07			X																					
SIZE	9144100	A	N	21										X														
LINE NAME	7941100	A	N	25										X														
PRICE UN BASE	8664130		N	06	2									X														
TAX FED EX UN	9464100		N	05	2									X														

System Identification 04 Process BILLING INPUT DESCRIPTION
 Set Identification 0ARD3 et Name SALES TAX FILE
 Frequency of Processing DAILY Volume: Average 120 Peak 200

Form Type MAGT. Page 5 of 23
 Source System I. D. 04 Prepared by: HCD
 Date 6/15/60

MANAGEMENT RULE NOS.																																									
ELEMENT NAME	ELEMENT CODE	ELEMENT DESCRIPTION					ELEMENT CLASS. (X)	NO. TIMES AN ENTRY MAY APPEAR ON THIS SET		VALIDATION RULE (S)	IDENTIFYING ELEMENT CODES (X)										Ref. Note (V)	REMARKS																			
		Suffix	Alpha	Numeric	Char.			Average	Peak		9464160																														
SALES TAX CODE	9464160	A	N	05			X																																		
SALES TAX RC	9464150	N		03	2			1	1		X																														

System Identification 04 Process BILLING INPUT DESCRIPTION

Set Identification 1201 at Name CUST FILE CHANGE

Form Type MANUAL

Prepared by: H LJ

Frequency of Processing WEEKLY Volume: Average 10 Peak 50

Source System I. D. 04

Date 6/15/60

MANAGEMENT RULE NOS.															013																	
ELEMENT NAME	ELEMENT CODE	ELEMENT DESCRIPTION			ELEMENT CLASS. (X)			NO. TIMES AN ENTRY MAY APPEAR ON THIS SET		VALIDATION RULE (S)	IDENTIFYING ELEMENT CODES (X)							Ref. Note (V)	REMARKS													
		Suffix	Alpha	Numeric	Char.	Recog.	Ident.	Action Mod.	Info.		Superfluous	Average	Peak	8981100	6813250																	
SALESMAN NO	8981100	N	05	0	X				1	1																						
CUST ACCT NO	6813250	N	04	0	X				1	1	X																					
DISC PC SUP BON	6920130	N	02	1		X			1	1	X	X																				
CUST NAME SOLD	6813200	A	70			X			1	1	X	X																				
CUST ADDR SOLD	6813210	A	N	70		X			1		X	X																				
POSTING IND	8760100	N	01	0			X		1	1	X	X																				

OUTPUT DESCRIPTION

System Identification 04

Process BILLING

Set Name INVOICE

Form Type MAT MASTER

Set Identification 1512B

Number Copies 8

Volume: Average 500

Peak 1000

Special Form I. D. 1510

Prepared by: HCD

Date 6/15/60

ELEMENT NAME	ELEMENT CODE	ELEMENT DESCRIPTION					Element Class. (X) Recog. Ident. Other	NO. TIMES AN ENTRY MAY APPEAR ON THIS SET		Set Type			SOURCE		IDENTIFYING ELEMENT CODES (X)										Ref. Note (V)															
		Suffix	Alpha	Numeric	Char.	Total		Decimal	Average	Peak	Process (X)	Retd. (A, B or X)	Action (X)	Source Set Identification for Direct Recording		8368100	9300100																							
ACCT CENTER	6017100	A			13								X	1512A			X																							
MO CUST ORD	8168620		N		02	0							X	1512A			X																							
DA CUST ORD	6836520		N		02	0							X	1512A			X																							
YR CUST ORD	9980620		N		02	0							X	1512A			X																							
ORDER NO	8368100	A	N		09					X				X	1512A			X																						
MO SHIPPED	8168650		N		02	0							X	1512A			X																							
DA SHIPPED	6836550		N		02	0							X	1512A			X																							
YR SHIPPED	9980650		N		02	0							X	1512A			X																							
INVOICE NO	7680100	A	N		09					X				X	1512A			X																						
CUST NAME SOLD	6813200		A		70					X			X	CARD1			X																							
CUST ADDR SOLD	6813210		A	N		70				X			X	CARD1			X																							
CUST NAME SHIP	6813220		A		70					X			X	1512A			X																							
CUST ADDR SHIP	6813230		A	N		70				X			X	1512A			X																							
CUST ORDER NO	6813100		A	N		12				X			X	1512A			X																							
SALESMAN NO	8981100		N		05	0				X			X	1512A			X																							
CUST ACCT NO	6813250		N		04	0				X			X	1512A			X																							
TERMS PAYMENT	9496100		A	N		23				X			X	1512A			X																							
SHIPG TERMS	9111100		A		23					X			X	1512A			X																							
CARRIER NAME	6466100		A		24					X			X	1512A			X																							
SHIP FROM TOWN	9110120		A		13					X			X	1512A			X																							
SALES TAX CODE	9464160		A	N		05				X			X	1512A			X																							
PRICE BASE	8664100		A	N		08				X			X	1512A			X																							
SIZE	9144100		A	N		21				X			X	CARD2			X																			X				
LINE NAME	7941100		A	N		25				X			X	CARD2			X																			X				

OUTPUT DESCRIPTION

System Identification 0A Process BILLING Set Name INVOICE Form Type MAT. MASTER
 Set Identification 1512B Number Copies 8 Volume: Average 500 Peak 1000 Special Form I. D. 1510

Prepared by: HCD
 Date 6/15/60

ELEMENT NAME	ELEMENT CODE	ELEMENT DESCRIPTION					Element Class (X)			NO. TIMES AN ENTRY MAY APPEAR ON THIS SET		SOURCE			IDENTIFYING ELEMENT CODES (X)										Ref. Note (V)										
		Suffix	Alpha	Numeric	Char.		Recog.	Ident.	Other	Average	Peak	Set Type	Process (X)	Retd. (A, B or X)	Action (X)	Source Set Identification for Direct Recording	8368100	9300100																	
					Total	Decimal																													
STOCK NUMBER	9300100	A	N	07				X	3	15			X	1512 A		X																			
QTY ORDERED	8768500		N	05	0			X	3	15			X	1512 A		X	X																		
QTY SHIPPED	8768550		N	05	0			X	3	15			X	1512 A		X	X																		
TAX FED EX UN	9464100		N	05	2			X	3	15		X		CARD2		X	X																		
UNIT PRICE	8664110		N	06	2			X	3	15	X					X	X																		
DISC PC QSA	6920110		N	02	1			X	1	1		X		1512 A		X																			
AMT QTY ALLOW	6116120		N	05	2			X	1	1	X					X																			
AMT PRKE EXT	6116100		N	07	2			X	3	15	X					X	X																		
AMT PRE TAX	6116180		N	08	2			X	1	1	X					X																			
AMT FED EX TAX	6116190		N	07	2			X	1	1	X					X																			
AMT CUST INY	6116220		N	08	2			X	1	1	X					X																			
SALES TAX PC	9464150		N	03	2			X	1	1	X					X																			
AMT SALES TAX	6116210		N	04	2			X	1	1	X					X																			

OUTPUT DESCRIPTION

System Identification 04 Process BILLING Set Name SALES DATA Form Type MAG. TAPE
 Set Identification 0401 Number Copies 1 Volume: Average 2000 Peak 5000 Special Form I. D. _____

Prepared by: HCD
 Date 6/15/60

ELEMENT NAME	ELEMENT CODE	ELEMENT DESCRIPTION					Element Class. (X)			NO. TIMES AN ENTRY MAY APPEAR ON THIS SET		Set Type			SOURCE	IDENTIFYING ELEMENT CODES (X)										Ref. Note (v)									
		Suffix	Alpha	Numeric	Char.		Recog.	Ident.	Other	Average	Peak	Process (X)	Retd. (A, B or X)	Action (X)		Source Set Identification for Direct Recording	9	3	0	1	2	3	4	5	6		7	8	9						
					Total	Decimal																													
STOCK NUMBER	9300100	A	N	07					1	1			X	1512 A																					
MO SHIPPED	81108650		N	02	0				1	1			X	1512 A																					
DA SHIPPED	6836550		N	02	0				1	1			X	1512 A																					
YR SHIPPED	9980650		N	02	0				1	1			X	1512 A																					
SALESMAN NO	8981100		N	05	0				1	1			X	1512 A																					
CUST ACCT NO	6813250		N	04	0				1	1			X	1512 A																					
QTY SHIPPED	8768550		N	05	0				1	1			X	1512 A																					
AMT PRICE EXT	6116100		N	07	2				1	1	X																								

OUTPUT DESCRIPTION

System Identification 01 Process BILLING Set Name INVENTORY DATA Form Type MAG TAPE
 Set Identification 0102 Number Copies 1 Volume: Average 2000 Peak 5000 Special Form I. D. _____

Prepared by: HCO
 Date 6/15/60

ELEMENT NAME	ELEMENT CODE	ELEMENT DESCRIPTION						Element Class. (X)		NO. TIMES AN ENTRY MAY APPEAR ON THIS SET		SOURCE			IDENTIFYING ELEMENT CODES (X)											Ref. Note (V)											
		Suffix	Alpha	Numeric	Char.		Recog.	Ident.	Other	Average	Peak	Set Type	Process (X)	Retd. (A, B or X)	Action (X)	Source Set Identification for Direct Recording			9300/00																		
					Total	Decimal																															
STOCK NUMBER	9300100	A	N		07					1	1			X						1512A																	
SHIP FROM TOWN	9110120	A			13					1	1			X						1512A				X													
QTY SHIPPED	8768550	N			05 0					1	1			X						1512A				X													

Appendix II, Page 14

OUTPUT DESCRIPTION

System Identification 040 Process BILLING Set Name ORDER ERRORS Form Type STANDARD PRINT
 Set Identification 040301 Number Copies 2 Volume: Average 5 Peak 10 Special Form I. D. _____

Prepared by: HCD
 Date 6/15/60

ELEMENT NAME	ELEMENT CODE	ELEMENT DESCRIPTION				Element Class: (X)	NO. TIMES AN ENTRY MAY APPEAR ON THIS SET		SOURCE			IDENTIFYING ELEMENT CODES (X)										Ref. Note (V)																				
		Suffix	Alpha Numeric	Char.			Average	Peak	Process (X)	Retd. (A, B or X)	Action (X)	Source Set Identification for Direct Recording																														
				Total	Decimal	Recog.	Ident.	Other																																		
ORDER NO	8368100	AN	09			X			1	1	X		1512A																													
SALES TAX CODE	9464160	AN	05				X		1	1	X																															

OUTPUT DESCRIPTION

System Identification 0A Process BILLING Set Name ORDER ERRORS Form Type STANDARD PRINT
 Set Identification 0A0302 Number Copies 2 Volume: Average 5 Peak 10 Special Form I. D. _____

Prepared by: HCD
 Date 6/15/60

ELEMENT NAME	ELEMENT CODE	ELEMENT DESCRIPTION						Element Class. (X)		NO. TIMES AN ENTRY MAY APPEAR ON THIS SET		Set Type Process (X) Retd. (A, B or X) Action (X)	SOURCE Source Set Identification for Direct Recording	IDENTIFYING ELEMENT CODES (X)										Ref. Note (V)																				
		Suffix	Alpha	Numeric	Char.		Recog.	Ident.	Other	Average	Peak			8368100																														
					Total	Decimal																																						
ORDER NO	8368100	A	N	09				X		1	1	X	1512A																															
SALESMAN NO	8981100		N	05	0			X		1	1	X										X																						
CUST ACCT NO	6813250		N	04	0			X		1	1	X										X																						

OUTPUT DESCRIPTION

System Identification 04 Process BILLING Set Name ORDER ERRORS Form Type STANDARD PRINT
 Set Identification 040303 Number Copies 2 Volume: Average 20 Peak 40 Special Form I. D. _____

Prepared by: HCD
 Date 6/15/60

ELEMENT NAME	ELEMENT CODE	ELEMENT DESCRIPTION				Element Class. (X)			NO. TIMES AN ENTRY MAY APPEAR ON THIS SET		Set Type Process (X) Retd. (A, B or X) Action (X)	SOURCE Source Set Identification for Direct Recording	8368100	IDENTIFYING ELEMENT CODES (X)										Ref. Note (V)				
		Suffix	Alpha Numeric	Char.		Recog.	Ident.	Other	Average	Peak																		
				Total	Decimal																							
ORDER NO	8368100	A	N	09		X			1	1	X	1512 A																
STOCK NO	9300100	A	N	07			X		1	1	X																	

OUTPUT DESCRIPTION

System Identification 04 Process BILLING Set Name ORDER ERRORS Form Type STANDARD PRINT
 Set Identification 04030A Number Copies 2 Volume: Average 5 Peak 10 Special Form I. D. _____

Prepared by: HCD
 Date 6/15/60

ELEMENT NAME	ELEMENT CODE	ELEMENT DESCRIPTION						Element Class. (X)	NO. TIMES AN ENTRY MAY APPEAR ON THIS SET		SOURCE			IDENTIFYING ELEMENT CODES (X)										Ref. Note (V)				
		Suffix	Alpha	Numeric	Char.		Recog.		Ident.	Other	Average	Peak	Set Type	Source Set Identification for Direct Recording	8368100	9300100												
					Total	Decimal																				Process (X)	Retd. (A, B or X)	Action (X)
ORDER NO	8368100	A	N	09			X		1	1	X	1512A																
STOCK NO	9300100	A	N	07			X		1	1	X	1512A	X															
QTY SHIPPED	8768550	N		05	0		X		1	1	X		X	X														

OUTPUT DESCRIPTION

System Identification 04 Process BILLING Set Name ORDER ERRORS Form Type STANDARD PRINT
Set Identification 046305 Number Copies 2 Volume: Average 5 Peak 10 Special Form I. D.

Prepared by: HCD
Date 6/15/60

ELEMENT NAME	ELEMENT CODE	ELEMENT DESCRIPTION				Element Class. (X)			NO. TIMES AN ENTRY MAY APPEAR ON THIS SET		SOURCE			IDENTIFYING ELEMENT CODES (X)												Ref. Note (V)																	
		Suffix	Alpha Numeric	Char.		Total	Decimal	Recog.	Ident.	Other	Average	Peak	Process (X)	Retd. (A, B or X)	Action (X)	Source Set Identification for Direct Recording	8368100	9300100																									
ORDER NO	8368100	AN		09					X		1	1			X	1512 A																											
STOCK NO	9300100	AN		07					X		1	1			X	1512 A																											
QTY ORDERED	8768500	N		050					X		1	1	X									X	X																				

OUTPUT DESCRIPTION

System Identification 04 Process BILLING Set Name ORDER ERRORS Form Type STANDARD PRINT
 Set Identification 040306 Number Copies 2 Volume: Average 2 Peak 5 Special Form I. D. _____

Prepared by: HCD
 Date 6/15/60

ELEMENT NAME	ELEMENT CODE	ELEMENT DESCRIPTION					Element Class. (X)			NO. TIMES AN ENTRY MAY APPEAR ON THIS SET			SOURCE			IDENTIFYING ELEMENT CODES (X)								Ref. Note (v)				
		Suffix	Alpha	Numeric	Char.		Recog.	Ident.	Other	Average	Peak	Set Type			Source Set Identification for Direct Recording	8368100	9300100											
					Total	Decimal						Process (X)	Retd. (A, B or X)	Action (X)														
ORDER NO	8368100	A	N	09			X		1	1			X	1512 A														
STOCK No	9300100	A	N	07			X		1	1			X	1512 A														
QTY SHIPPED	8768550			05	0		X		1	1	X				X	X												
QTY ORDERED	8768500			05	0		X		1	1	X				X	X												

OUTPUT DESCRIPTION

System Identification OA Class BILLING Set Name ORDER ERRORS Form Type STANDARD PRINT

Set Identification 040307 Number Copies 2 Volume: Average 5 Peak 10 Special Form I. D.

Prepared by: HCD
Date 6/15/60

ELEMENT NAME	ELEMENT CODE	ELEMENT DESCRIPTION					Element Class. (X)	NO. TIMES AN ENTRY MAY APPEAR ON THIS SET		Set Type			SOURCE	IDENTIFYING ELEMENT CODES (X)											Ref. Note (✓)					
		Suffix	Alpha	Numeric	Char.			Recog.	Ident.	Other	Average	Peak	Process (X)	Retd. (A, B or X)	Action (X)	Source Set Identification for Direct Recording	8368100													
					Total	Decimal																								
ORDER NO	8368100	A	N	09			X		1	1			X	1512A																
MO SHIPPED	8168650		N	02	0		X		1	1	X							X												
DA SHIPPED	6836550		N	02	0		X		1	1	X							X												
YR SHIPPED	9980650		N	02	0		X		1	1	X							X												

OUTPUT DESCRIPTION

System Identification 0A Process BILLING Set Name CUST CHANGE ERROR Form Type STANDARD PRINT
 Set Identification 0A0A Number Copies 2 Volume: Average 1 Peak 10 Special Form I, D. _____

Prepared by: HCD
 Date 6/15/60

ELEMENT NAME	ELEMENT CODE	ELEMENT DESCRIPTION				Element Class. (X)			NO. TIMES AN ENTRY MAY APPEAR ON THIS SET		SOURCE			IDENTIFYING ELEMENT CODES (X)								Ref. Note (V)				
		Suffix	Alpha Numeric	Char.		Recog.	Ident.	Other	Average	Peak	Set Type			Source Set Identification for Direct Recording	8981100	6813250										
				Total	Decimal						Process (X)	Retd. (A, B or X)	Action (X)													
SALESMAN NO	8981100		M	05	0	X			1	1			X	1201												
CUST ACCT NO	6813250		M	0A	0	X			1	1			X	1201	X											
DISC PC SUP BON	6920130		M	02	1		X		1	1	X				X	X										
CUST NAME SOLD	6813200		A	70			X		1	1	X				X	X										
CUST ADDR SOLD	6813210		AM	70			X		1	1	X				X	X										
POSTING IND	8760100		M	01	0		X		1	1	X				X	X										
ERROR REASON	7000100		A	09			X		1	1	X				X	X										

PROCESS DESCRIPTION

ge 19 Of 23

System Identification 04

Prepared by HCD

Process BILLING Validation

Date 6/15/60

LINE NO.	MANAGEMENT RULE NO.		SOURCE				COND.					SOURCE/DISPOSITION				MANAGEMENT RULE SUFFIX AND FREQUENCY																								
	C/A	Current	Prior	ELEMENT Name, Prior Result or Actual Value	SUF.	Set Ident. or Rule No. for Prior Result	Greater Than	Less Than	Equal To	Oper (+ - / X I)	Set = To (V)	ELEMENT Name, Result, Prior Result or Actual Value	SUF.	Set Ident. or Rule No. for Prior Result	Oper (+ - / X)	Note Ref. (V)	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R						
001	C	001		MO SHIPPED		1512 A	√	√			(01)				1	Y	N																							
002							√	√			(12)					Y	N																							
003	C			DA SHIPPED		1512 A	√	√			(01)					Y		N																						
004							√	√			(31)					Y			N																					
005	C			YR SHIPPED		1512 A	√	√			/CURRENT YR/					Y				N																				
																5	1	1	1	1	1																			
006	A			MO SHIPPED		1512 A				V	MO SHIPPED		040307			X	X	X	X	X																				
007	A			DA SHIPPED		1512 A				V	DA SHIPPED		040307			X	X	X	X	X																				
008	A			YR SHIPPED		1512 A				V	YR SHIPPED		040307			X	X	X	X	X																				
009	A			/CURRENT MO/						V	MO SHIPPED		1512 A			X	X	X	X	X																				
010	A			/CURRENT DA/						V	DA SHIPPED		1512 A			X	X	X	X	X																				
011	A			/CURRENT YR/						V	YR SHIPPED		1512 A			X	X	X	X	X																				

PROCESS DESCRIPTION

System Identification 0A

Prepared by HCD

Process BILLING Validation

Date 6/15/60

LINE NO.	C/A	MANAGEMENT RULE NO.		SOURCE				COND.		SOURCE/DISPOSITION				Oper (+ - / X) None Ref. (V)	MANAGEMENT RULE SUFFIX AND FREQUENCY																					
		Current	Prior	ELEMENT		Set Ident. or Rule No. for Prior Result	Greater Than	Less Than	Equal To	Oper (+ - / X)	Set = To (V)	ELEMENT			Set Ident. or Rule No. for Prior Result	Cond: Y = Condition Is Satisfied N = Condition Is Not Satisfied Blank = Condition Not Applicable Action: X = Action to Be Taken Blank = Action Not to Be Taken	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R		
				Name, Prior Result or Actual Value	SUF.							Name, Result, Prior Result or Actual Value	SUF.																							
012	C	002		SALES TAX CODE		1512 A		✓			SALES TAX CODE		0ARD3	1	Y	N																				
																5																				
013	A			SALES TAX CODE		1512 A				✓	SALES TAX CODE		0A0301			X																				
014	A			(x & b & b)						✓	SALES TAX CODE		1512 A			X																				
015	C	003		SALESMAN NO		1512 A		✓			SALESMAN NO		0ARD1	2	Y	N																				
016				CUST ACCT NO		1512 A		✓			CUST ACCT NO		0ARD1		Y	N																				
																5																				
017	A			SALESMAN NO		1512 A				✓	SALESMAN NO		0A0302			X	X																			
018	A			CUST ACCT NO		1512 A				✓	SALESMAN NO		0A0302			X	X																			
019	C	004		STOCK NO		1512 A		✓			STOCK NO		0ARD2	2	Y	N																				
																5																				
020	A			STOCK NO		1512 A				✓	STOCK NO		0A0303			X																				
021	C	005		QTY SHIPPED		1512 A		✓			/NNNNN/			2	Y	N	Y	Y																		
022	C			QTY ORDERED		1512 A		✓			/NNNNN/				Y	Y	N	Y																		
023	C			QTY ORDERED		1512 A	✓	✓			QTY SHIPPED		1512 A		Y		N																			
																5																				
024	A			QTY SHIPPED		1512 A				✓	QTY SHIPPED		0A030A			X																				
025	A			QTY ORDERED		1512 A				✓	QTY ORDERED		0A0305				X																			
026	A			QTY SHIPPED		1512 A				✓	QTY SHIPPED		0A0306																							
027	A			QTY ORDERED		1512 A				✓	QTY ORDERED		0A0306																							

Appendix II, Page 24

PROCESS DESCRIPTION

System Identification 04

Prepared by HCD

Process BILLING

Validation

Date 6/15/60

LINE NO.	C/A	MANAGEMENT RULE NO.		SOURCE			COND.		SOURCE/DISPOSITION			Note Ref. (V)	MANAGEMENT RULE SUFFIX AND FREQUENCY																							
		Current	Prior	ELEMENT	Set Ident. or Rule No. for Prior Result	SUF.	Greater Than	Less Than	Equal To	Oper (+ - / X %)	Set = To (V)		ELEMENT	Set Ident. or Rule No. for Prior Result	SUF.	Y	N	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R		
																																			Name, Prior Result or Actual Value	Name, Result, Prior Result or Actual Value
043	C	011	008	SALES TAX CODE	1512 A						(X &&&&)			Y	N																					
														5	3																					
044	A			SALES TAX PC	0ARD3				X		AMT PRE TAX	1512 B																								
045											AMT SALES TAX	1512 B			X																					
046											RESULT A				X																					
047				ZERO							RESULT A			X																						
048				SALES TAX PC	0ARD3						SALES TAX PC	1512 B			X																					
049	A	012	011	RESULT A	(010)						RESULT A	(007)																								
050			010	RESULT A	(011)						AMT CUST INY	1512 B																								
			007																																	

PROCESS DESCRIPTION

System Identification 04

Prepared by HCD

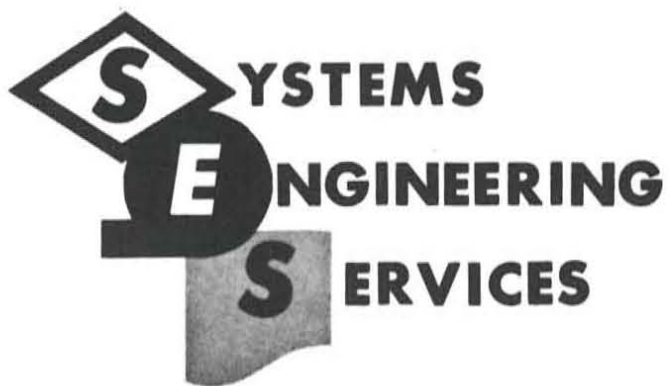
Process BILLING Validation

Date 4/15/60

LINE NO.	MANAGEMENT RULE NO.		SOURCE				COND.		SOURCE/DISPOSITION				MANAGEMENT RULE SUFFIX AND FREQUENCY																													
	C/A	Current	Prior	ELEMENT	Set Ident. or Rule No. for Prior Result	SUF.	Greater Than	Less Than	Equal To	Oper (+ - / X Z)	Set = To (V)	ELEMENT	Set Ident. or Rule No. for Prior Result	SUF.	Oper (+ - / X)	Note Ref. (V)	Cond: Y = Condition Is Satisfied N = Condition Is Not Satisfied Blank = Condition Not Applicable Action: X = Action to Be Taken Blank = Action Not to Be Taken																									
																	Name, Result, Prior Result or Actual Value	Name, Result, Prior Result or Actual Value	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R						
051	C	013		SALESMAN NO	1201		✓				SALESMAN NO	0ARDI					Y	Y	N	N	N																					
052				CUST ACCT NO	1201		✓				CUST ACCT NO	0ARDI					Y	Y	N	N	N																					
053				POSTING IND	1201		✓				(1)						Y		Y	Y																						
054							✓				(2)							Y		Y	Y																					
055							✓				(3)								Y		Y	Y																				
056	A			/BLANK/						✓							X																									
057					1201		✓														XX																					
058				DISC PC SUP BON	1201		✓				DISC PC SUP BON	0ARDI						X																								
059				DISC PC SUP BON	1201		✓				DISC PC SUP BON	0A0A					X	XX		XX																						
060				CUST NAME SOLD	1201		✓				CUST NAME SOLD	0A0A					X	XX																								
061				CUST ADDR SOLD	1201		✓				CUST ADDR SOLD	0A0A					X	XX																								
062				(MATCHED)			✓				ERROR REASON	0A0A					X																									
063				(UNMATCHED)			✓				ERROR REASON	0A0A							XX	XX																						

STANDARD REFERENCE NOTES FOR VALIDATIONS

<u>Note Number</u>	<u>Explanation</u>
1.	If the element is not valid, continue with the execution of the Management Rules for validations and processes indicated by the set that contains the invalid element.
2.	If the element is not valid, continue with the execution of the Management Rules for validations indicated by the set that contains the invalid element. Do not execute the Management Rules for processes indicated by the set that contains the invalid element.
3.	If the element is not valid, do not continue with the execution of the Management Rules for validations. Do not execute the Management Rules for processes.



CLEARINGHOUSE REPORT

AN INSURANCE COMPANY
FILE MAINTENANCE PROBLEM

June 10, 1961
Ref. No. 1F3

Burton Grad

INTERNATIONAL BUSINESS MACHINES CORPORATION
White Plains, New York

AN INSURANCE COMPANY FILE MAINTENANCE PROBLEM

Burton Grad
IBM

This report presents two methods (flow chart and decision table) for representing the decision logic of a complex problem; it thereby provides a means of comparing the relative merits of the two techniques. Some considerations in such a comparison are: clarity of understanding, ease of modifying, ability to detect logical errors and omissions, ability to see important relationships, etc. This example by no means represents a controlled test or evaluation of flow charts vs. decision tables; it has, however, provided some insight and firsthand experience with the two methods on an identical problem.

The particular problem is concerned with master file maintenance and controlling key operating procedures of a large insurance company. The operations are presented at the systems level and while not precise enough for direct coding, should be accurate and structurally sound. Some of the logical inaccuracies that exist in the flow chart were corrected in the decision tables.

With the problem solution initially represented in flow chart form, it was then decided to explore the capability of decision tables for describing such a complex decision procedure. It took approximately 25 man-hours to study the flow chart, understand and structure the problem, and prepare the decision tables. This short time did not allow thorough review and debugging of the decision tables. The most difficult task was to understand the problem from the information available; considerable time and effort were required with the flow chart originator toward this end. However, once the flow chart was grasped, the problem could be subdivided into several major portions. It seemed at the time that this might be one main advantage of tables, i. e., they seem to force logical structure.

The Basic Problem Solution

A series of insurance policies are maintained in a master random access file; because of the large number of these policies, and the relative infrequency of change or use, it is desirable to have a dictionary track which contains a brief record for each policy. The overall control table (001) is concerned with detecting which policies need to be acted upon while checking each of the policy summary records in sequence.

Another complication of the job is that a single customer may have a multiple account, i. e. , more than one policy; if a customer has a multiple account, he may go on a monthly pay plan instead of the normal three-payment or one-payment method. Tables 002, 003, and 012 detect and handle multiple account and monthly pay cases.

There are two major types of activities to the file. The first is that which is scheduled because of the date, such as renewals, terminations, etc. This is handled by Tables 005 and 009, which are concerned with scheduled activities. The other type of work involves handling transactions, where a change in policy status or introduction of a new policy takes place. Tables 008 and 011 take care of transaction activities.

The remaining table, Table 010, is a closed procedure table which is used in a variety of cases to write out a previous policy and obtain the policy to be examined.

Attached you will find the flow chart used to describe the insurance company job, followed by the decision tables which cover the same ground.

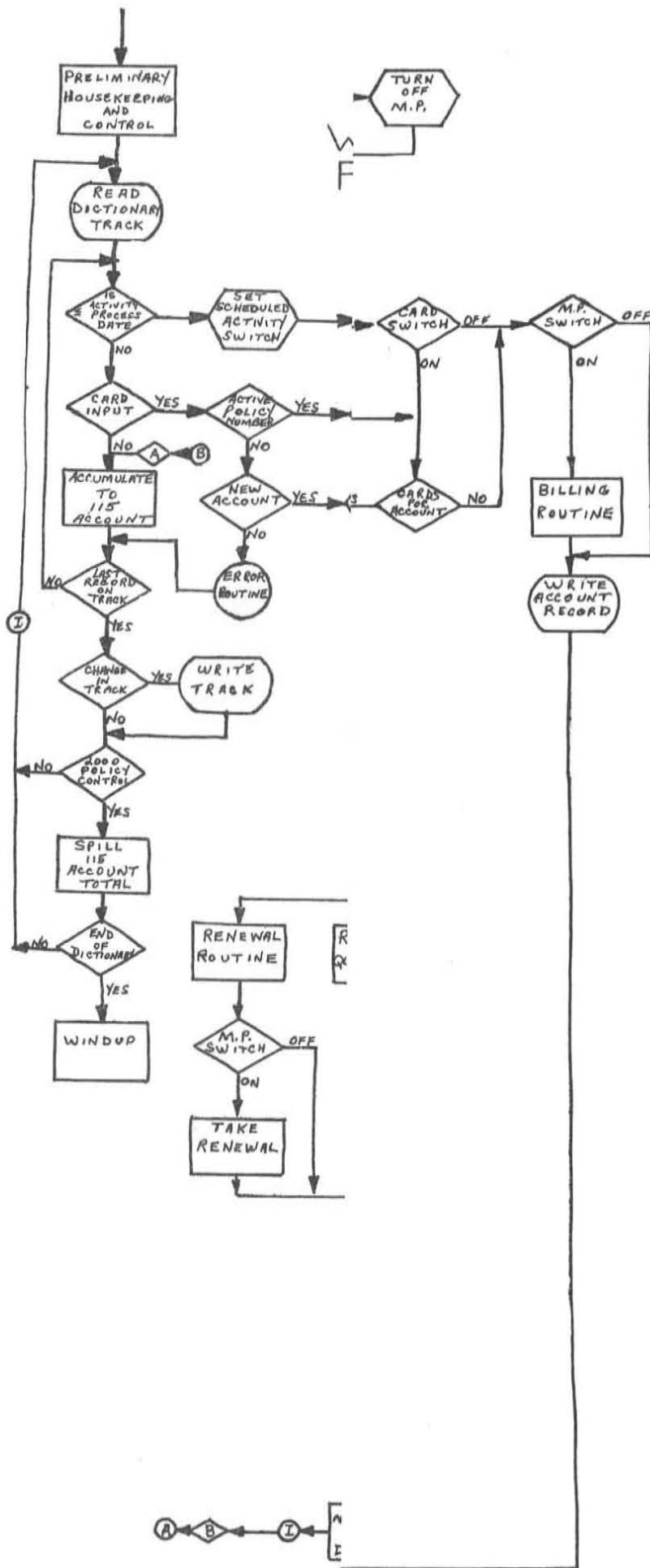


TABLE 001 Overall Control

Rule No.	1	2	3	4	5	6	7	8	9	10	11	12	13
START	Y	N	N	N	N	N	N	N	N	N	N	N	N
Activity Date ≤ Process Date		N	N	N	N	N	N	N	N	N	Y	Y	N
Card Input for this account		N	N	N	N	N	N	N	Y	Y	N	Y	Y
Last Record on Track		N	Y	Y	Y	Y	Y	Y					
Change Track Switch ON			N	N	N	Y	Y	Y					
Policy Control No. = 2000			N	Y	Y	N	Y	Y					
End of Dictionary				N	Y		N	Y					
Active Policy No.									N	N			Y
New Account									N	Y			
Preliminary Housekeeping & Control	X												
Error... Not Active Policy									X				
Set Status Code =	0				2			2					
Set Track Change Switch ON										X	X	X	X
Accumulate to 115 Account		X	X	X	X	X	X	X					
Write Dictionary Track						X	X	X					
Set Track Change Switch OFF						X	X						
Spill 115 Account Total				X	X		X	X					
Read Next Dictionary Track	X		X	X		X	X						
Setup Next Dictionary Item	X	X	X	X		X	X						
Set Policy Control No. =	100		+100	100		+100	100						
Windup (incl. Table 010)					X			X					
Read Input Card	X								X				
Set Card Switch ON										X		X	X
Set Schedule Activity Switch ON											X	X	
GO TO TABLE	001	001	001	001	Stop	001	001	Stop	001	002	002	002	002

TABLE 002 Special Processing

Rule No.	1	2	3	4	5	6	7	8	9	10	11	12	13
Multiple Account	N	N	N	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y
Monthly Pay	N	N	N	N	N	N	Y	Y	Y	Y	Y	Y	Y
Card Switch ON	Y	N	Y	N	Y	Y	N	N	Y	Y	Y	Y	Y
Type of Transaction = Status Change	N		Y		N	Y			N	N	Y		
= Cash									N	N		Y	Y
Transaction Switch = C							N	Y	N	Y		N	Y
Set Multiple Account Switch ON				X	X	X	X		X		X	X	
Set Monthly Pay Switch ON							X		X		X	X	
Read Multiple Account Record				X	X		X		X			X	
Setup Multiple Account Tally & Control				X	X		X	X	X	X			
Do Policy Record Setup (Table 010)	X	X											
Set Transaction Switch =								b		b		"C"	
Handle Monthly Pay Cash Transaction												X	X
Read Input Card												X	X
GO TO TABLE	008	005	012	003	003	012	003	003	003	003	012	011	011

TABLE 003 Multiple Account Control

Rule No.	1	2	3	4	5	6	7	8	9	10	11
End of Account	Y	Y	Y	Y	N	N	N	N	N	N	N
Card Switch ON	N	N	Y	Y	Y	Y	Y	Y	Y	N	N
Cards for Tally					High	High	Low	Low	EQ		
New Business			Y	N			Y	N			
Monthly Pay Switch ON	N	Y									
Schedule Activity					N	Y				N	Y
Error... Card out of line				X				X			
Set Transaction Switch =				"L"				"L"			
Change Tally			X				X				
Set Schedule Activity Switch OFF	X	X									
Billing Routine		X									
Write Account Record	X	X									
Tally					X					X	
Read Input Card				X				X			
Do Policy Record Setup (Table 010)			X			X	X		X		X
Set Multiple Account Switch OFF	X	X									
GO TO TABLE	005	005	008	011	003	005	008	011	008	003	005

TABLE 005 Scheduled Activity

Rule No.	1	2	3	4	5	6	7	8	9	10	11	12	13	14
Schedule Activity Switch ON	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	N	N	N
Monthly Pay Switch ON	Y	N												
Multiple Account Switch ON	Y	Y	N	Y	N	Y	N	Y	N	Y	N		Y	N
Type of Activity = Renewal	Y	Y	Y											
= Renewal Questionnaire				Y	Y									
= Termination						Y	Y							
= Cancellation								Y	Y					
= Bills & Reminders										Y	Y			
Transaction Switch = "T"												N	Y	Y
Renewal Routine	X	X	X											
Renewal Questionnaire Routine				X	X									
Termination Routine						X	X							
Cancellation Routine								X	X					
Bills & Reminders Routine										X	X			
Zero Record & Open Address						X	X							
Take Renewal	X													
Compute New Act. Date	X	X	X	X	X	X	X	X	X	X	X			
Write Policy Record	X	X	X	X	X	X	X	X	X	X	X		X	X
Insert New Act. Date in Dictionary			X		X		X		X		X	X		X
Tally	X	X		X		X		X		X			X	
Change Multiple Account Record	X	X		X		X		X		X			X	
Set Schedule Activity Switch OFF			X		X		X		X		X			
GO TO TABLE	003	003	001	003	001	003	001	003	001	003	001	001	003	001

TABLE 008 Transaction Activity

Rule No.	1	2	3	4	5	6	7	8	9	10	11	12
Card Switch ON	Y	Y	Y	N	N	N	N	N	N	N	N	N
Transaction Type = New Business = Endorsement	Y		N	Y		N				N		
Control Code = Policy Nos. match		Y	N		Y	N					N	Y
New Business Preliminary	X			"NB"	"EN"	"RT"	"NB"	"EN"	"RT"	"NB"	"EN"	"RT"
Endorsement Preliminary		X		Y	Y	Y	N	N	N	Y	Y	Y
Handle Transaction			X			X						
Process Card	X	X		X	X							
Read Input Card	X	X	X	X	X	X						
Set Card Switch OFF	X	X	X									
Set Card Switch ON										X	X	X
Set Control Code =	"NB"	"EN"	"RT"									
Finish Endorsement								X			X	
Finish New Business							X			X		
Compute New Activity Date							X	X	X			
GO TO TABLE	008	008	008	008	008	008	009	009	009	008	008	008

TABLE 009 Scheduled Activity Check

Rule No.	1	2	3	4	5
Schedule Activity Switch ON	Y	Y	Y	N	N
Old Schedule Necessary	Y	N	N		
Analysis of Activity Change		Y	N	Y	N
Set Schedule Activity Switch ON				X	
Set Schedule Activity Switch OFF			X		
Set Transaction Switch = "T"			X		X
GO TO TABLE	005	005	005	005	005

TABLE 010 (DO) Policy Record Setup

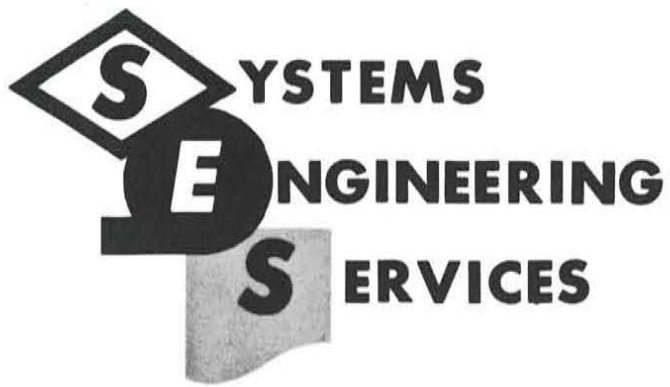
Rule No.	1	2	3
Status Code	1	0	2
Seek Policy Record	X	X	
Pre Activity Write	X		X
Activity Write	X		X
Read Policy Record	X	X	
Set Status Code =		1	

TABLE 011 Card Input Control

Rule No.	1	2	3	4	5
Card Input for this Policy	Y	Y	N	N	N
Transaction Switch =	"C"	"L"	"C"	"C"	"L"
Schedule Activity Switch ON			Y	N	
Set Card Switch OFF			X	X	X
GO TO TABLE	002	003	002	003	003

TABLE 012 Status Change Control

	Rule No.	1	2	3	4	5
Multiple Account		Y	Y	Y	N	N
Monthly Pay			Y	N	N	N
Type of Status Change = On Multiple Account					Y	
= On Monthly Pay				Y		
= On Multiple Account & Monthly Pay						Y
= Off Multiple Account		Y				
= Off Monthly Pay			Y			
Read Multiple Account Record		X	X	X		
Setup & Write Multiple Account Record					X	X
Change Dictionary Track this item		X	X	X	X	X
Change Multiple Account Record		X	X	X		
Set Transaction Switch =			"C"	"C"		
Read Input Card			X	X	X	X
Set Type of Transaction = blank		X				
GO TO TABLE		002	011	011	002	002



CLEARINGHOUSE REPORT

TABLES SIGNAL BETTER
COMMUNICATION

June 1, 1961
Ref. No. 1F1

Burton Grad

INTERNATIONAL BUSINESS MACHINES CORPORATION

White Plains, New York

Talk Given by Burton Grad, Manager IBM Systems Engineering Development

The pilot is preparing to land his single engine plane at the airport; it is late at night and his fuel supply is low. He calls to the radio tower and asks for landing instructions. All he hears in return is a babble in a foreign language which he can't understand.

The executive has spent the last hour of his day dictating an important speech; the next morning he comes in and wants to review the material. His secretary is out ill. The other girls in the office all read Gregg, not Pitman.

A design engineer has carefully prepared a number of complex Boolean equations to explain the operation of a new computer circuit. He shows these to the manufacturing engineer to give an indication of what needs to be constructed. The manufacturing engineer says, "I don't understand Boolean algebra."

We could go on and on citing examples like these of events and occurrences where lack of a common language for communication causes difficulties ranging all the way from the most trivial to the deadly. Systems Engineering faces communication barriers as serious as those of any profession. The systems engineer today does not have a language to communicate with management; he does not have a language to communicate with computer programmers; he does not have a language to communicate with functional specialists; he does not even have a language to communicate with other systems engineers.

Programmers who have learned one computer at the machine language level can't understand the programming of another machine at the machine language level without spending the time necessary to learn the second machine's special codes and instructions. For this reason (among others) there has been intensive effort to develop common languages like FORTRAN, Commercial Translator and COBOL which will be applicable to a number of machines. But the communication between programmer and machine is merely a small part of the total problem.

For Systems Engineering it is vital to develop tools and techniques to permit a manager to state his decision criteria and decision rules. We must find a common language so systems engineers can communicate with product engineers, accountants, and manufacturing planners, to find out their decision rules and decision logic; that is critical to determine the characteristics of the system that is going to be modelled or controlled. A method must be found for two-way communication with computer programmers to be sure that the intended decision rules are in fact being executed. A technique is needed to aid the systems engineer in establishing complete decision rules and in predetermining that these rules will accomplish the intended goals.

In the past, this problem has not been as severe. Because of the limited size of business systems problems, we could depend on the programmer to understand the particular problems well enough to be sure the logic was correct and to check the problem out thoroughly. However, as the systems we are trying to solve become larger and more complex, this expedient is no longer satisfactory. Systems engineers must take on the responsibility for designing the decision logic and for insuring that it is being executed properly. To do this systems engineers must have a professional language which will serve for effective intercommunication.

What has caused the communication void? What has caused this communication moat surrounding the systems engineer? There are at least three major factors involved:

- 1- The inability to clearly and concisely express decision logic and decision rules for describing business systems.
- 2- The inability to show cause-effect relationship between conditions and actions.
- 3- The inability to guarantee or even aid in achieving logical completeness in establishing decision rules.

Today, we have available a number of techniques which have been applied to solving the communication problem: we've tried to use narrative, flow charts and even logical equations. But none of these has filled the bill. Each has major drawbacks; the failure of these known techniques has led to consideration of another alternative: decision tables.

Decision Tables

Decision tables are a formal method for describing decision logic in a two-dimensional display. The layout clearly shows the cause and effect relationship between conditions and actions; it explicitly relates decision alternatives.

Decision tables use a format which is familiar to us from analytical, financial, and statistical tables. Since the days of the Babylonians, people have used tables as a means of organizing information where the relationships were complex or the amount of data great. These data tables appear to be superior to many other forms of information organization because:

- 1- They provide clarity and conciseness through data classification.
- 2- They clearly show relationship of dependent to independent variables.
- 3- They explicitly indicate omissions.

Decision tables use tabular format to represent dynamic situations. Where we have used flow charts, narrative, or logical equations to describe decision logic, or an operating procedure, we now find it possible to use decision tables for these same jobs. The argument in favor of tables is their relative convenience and effectiveness, not that they can describe systems that cannot also be described in other ways.

Tabular form has been used by programmers since the earliest days of computers. The most common use of tables has been to relate some function to an argument. Given the value of one factor, the table provides the value of another dependent factor. For example, a table might relate capitals to states (Figure 1). Given the state name, determine the name of the capital.

STATE	Alabama	Alaska		Wyoming
CAPITAL	Montgomery	Juneau		Cheyenne

In this example State appears above the double line and Capital below; each different state name is in a column and physically below it, the name of the corresponding capital. If the State is Alabama, then the Capital is Montgomery; if the State is Alaska, then the Capital is Juneau.

An extension of this concept is seen in Figure 2 in the use of a matrix to display the value of a particular factor as a function of multiple variables.

HEALTH	EXCELLENT	GOOD	FAIR	POOR
AGE				
25 < 30	1.27	1.62	1.90	2.73
30 < 45	1.83	2.12	2.53	3.42
45 < 55	2.51	2.93	3.47	5.27
55 < 65	3.29	3.91	4.85	8.73
65 < 70	5.21	6.45	7.61	10.87

Insurance premium rates are shown as a function of health and age. In the example, if health is excellent and age is between 25 and 35, then the rate is \$1.27. However, if health is poor and age between 55 and 65, then the rate is \$8.73. Unfortunately, the visual effectiveness of a matrix is reduced when the number of independent variables exceeds two or the number of dependent variables is greater than one.

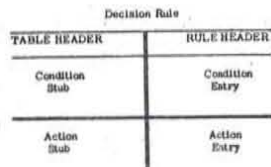
Because of the natural benefits from using tables, it seems that there should be some way to generalize tabular form so that any number of independent and dependent variables might be shown with clear visual correspondence. Figure 3 (on the next page) shows a table with four independent and three

dependent factors where clarity, interrelationship and comprehensiveness have been maintained.

Health	Excellent	Excellent	Poor
Age	25, 35	20, 35	65
Section of Country	East	East	West
Sex	Male	Female	Female
Premium Rate	1.27	1.18	9.82
Policy Limit	300,000	100,000	10,000
Type of Policy	A, B, or C	A, B, or C	R

In this example, the decision table indicates insurance premium rate, policy limit, and type of policy as a function of health, age, section of country, and sex. If the applicant is in excellent health, between 25 and 35 years of age, from the East, and is a male, his rate is \$1.27, the insurance limit is \$200,000, and he may be issued policy type A, B, or C. All of the alternatives are clearly set forth, one by one, across the table.

To obtain a better understanding of a decision table, let's look at its fundamental elements as shown in Figure 4.



The double lines serve as demarcation: CONDITIONS are shown above the horizontal double lines, ACTIONS below. The STUB is to the left of the vertical double line, ENTRIES to the right. A condition states a relationship. An action states a command.

If all the conditions in a column are satisfied then the actions in that column are executed. Each such vertical combination of conditions and actions is called a RULE. In the same column with the entries for each rule, there may be specialized data relating to that rule; this is called the RULE HEADER. Similarly, each table may have certain specialized information which is called the TABLE HEADER.

Consider another sample table which contains all the same elements, but has some different properties. This table is Figure 5.

TABLE: CREDIT	Rule 1	Rule 2	Rule 3	Rule 4
Credit limit is O.K.	Y	N	N	N
Pay experience is favorable		Y	N	N
Special clearance is obtained			Y	N
Approve order	X	X	X	
Return order to Sales				X

The first rule would be read: If credit limit is OK, then approve order. The second rule would be read: If credit limit is not OK and pay experience is favorable, then approve order. In this LIMITED ENTRY table, the entire condition or action must be written in the stub. The condition entry is limited to indicating whether the corresponding condition should be asserted, negated or ignored; the action entry indicates if the action stub should be executed or ignored.

This is in contrast, as you may note, to the table shown in Figure 3, which is called an EXTENDED ENTRY table. In this case the individual condition or action information extends from the stub into the corresponding entries. In any given table, we can, of course, mix extended and limited entry form, whichever is more convenient for a particular condition or action.

The Use of Decision Tables

To this point sample decision tables and their elements have been discussed to describe concept and structure. Now the application and use of decision tables will be presented. A number of experiments conducted over the past four years have used decision tables on a variety of problems; these will be reviewed briefly.

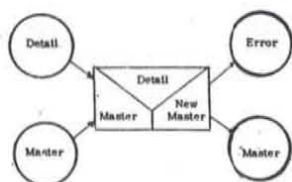
While I was project leader for General Electric's Integrated Systems Project, the potential application of tables to a wide variety of problems was explored including its use for product design, operation planning, cost determination, factory scheduling, etc. The results certainly revealed the opportunity of using decision tables as a major new tool to clarify communication among different technical specialists as well as between these specialists and computer programmers. It was stimulating to watch a manufacturing engineer suddenly grasp product design decision logic and then point out where restraints had been introduced by the product engineer that were of little value to anybody. Through this kind of examination, significant improvements might be made in the total product.

At Sutherland Company, a consulting firm in Peoria, Illinois, management decision rules have been studied with various customers and expressed in tabular form. These decision tables have been applied to Air Force logistics and various commercial situations such as accounts receivable, accounts payable, etc. From all reports, this work has permitted a more effective and comprehensive statement of the current decision logic and provided more meaningful and understandable communication between systems men and programmers.

An area of experimentation already familiar to many of you is the work done at Hunt Foods and Industries by Mr. O. Y. Evans, who is now with IBM. Mr. Evan's work was directed toward communication among different systems men, and from systems men to programmers, concerning the complex decision rules involved in stock control, sales analysis, etc. The results demonstrate that this approach was an effective formal way to state very complex logic without requiring knowledge of Boolean algebra or any other precise mathematical technique.

IBM has been working with several of its customers investigating potential applications of decision tables to a wide variety of problems. From these experiments, it seems clear that decision tables are frequently easier to prepare than comparable programming methods, and that they are an effective aid to systems analysis. In these experiments, communication between systems engineer and programmer has been substantially improved; communication between systems engineer and management has also benefitted from the common description of decision rules.

To convey how tables can be developed, let's follow the process through the significant problem of file maintenance. The block diagram in Figure 6 indicates the essential elements of the problem solution.



A detail file and a master file are the two inputs. The updated master file and an error file are the principal outputs. Within the computer, three basic areas are assigned: master, detail, and new master. The purpose of the update logic is to modify the incoming master file by the detail information to produce an updated master file containing any additions and changes and from which deleted records have been eliminated.

Figure 7 (on the following page) is one of two tables prepared to perform this job.

Rule 1 states the starting condition. At the start, one master record and one detail record are read into the corresponding memory areas. At this point, sequence control returns to the beginning of the table.

Rule 2 and all the following ones are now pertinent. Rule 2 specifically handles the end of job conditions, i.e., end of detail and end of master. In this case, control is transferred to End, a closing routine to provide for sentinels, tape marks, etc.

TABLE: Update	Rule #	01	02	03	04	05	06	07	08
Start		Y	N	N	N	N	N	N	Else
End of detail			Y	Y	N	N	N	N	
End of Master			Y	N	Y	N	N	N	
Detail						Master	Master	*Master	
Detail an "addition"					Y	Y			
Do Error Routine									x
Move Master to New Master									x
Move Detail to New Master					x	x			
Set Addition Switch					On	On		Off	
Write Master				x			x		
Read Master		x		x			x		
Read Detail		x			x	x			x
Go to Table		Update	End	Update	Change	Change	Update	Change	Update

Rule 3 describes the situation when the end of detail has been reached, but not the end of master. Since there can be no further changes, additions, or deletions to the original master, the actions are to write the updated master from the master area, read another master, and then return to the beginning of the table.

In Rule 4, the end of master has been found, but not the end of detail; the remaining details should only be additions. Therefore, the information in the detail area is moved to the new master area, the addition switch is set on, a new detail record is read, and control transferred to the Change Table.

Rules 5, 6, and 7 are concerned with cases where neither the detail nor the master file has ended. The identification number in the detail area is compared to the identification number in the master area. Rule 5 considers the event when the detail is less than the master; in this case the detail should be an addition in order to follow the same logic of Rule 4. In Rule 6 the detail is greater than the master; consequently the same logic as Rule 3 applies. Rule 7 covers the case where master and detail are equal. The information in the master area is moved to the new master area, and control is transferred to the Change Table.

The final rule, Rule 8, is the ELSE Situation. When this occurs something has gone wrong, since all legitimate possibilities have already been examined. An error routine is carried out; then another detail record is read. Rule 8 will take care of cases involving sequence errors in the master file and certain types of sequence errors in the detail file (if the out-of-sequence detail is not an addition). It will also take care of any non-matching detail which is not an addition.

The table can be rearranged to aid programming efficiency: columns with higher frequency of success should be moved to the left and those with lower frequency to the right. Rules 1 and 2 would be way over to the right since they occur only once in each program. Depending upon the particular data, Rule 6 (the column where the detail is greater than the master) will probably be the most frequent case and should be the first one considered. One recommended order is: 6, 7, 5, 3, 4, 1, 2, 8.

Another concept for improving program efficiency is to rearrange the conditions to present the most discriminating condition at the top and the least discriminating at the bottom. For example, the start condition, which is shown first, probably should be last since this only distinguishes one case out of all the thousands that will occur. A similar statement can be made about end of detail and end of master. It seems evident that the comparison of detail to master would be the most discriminating criteria and therefore placed first in the table.

The Case for Tabular Form

Look once more at Figure 7 and compare its statement of the update decision logic with that in the following narrative. Which is clearer and more concise, which shows cause-effect relationships better, which aids more in determining logical completeness.

Mr. T. F. Kavanagh speaking at the 1960 Eastern Joint Computer Conference had this to say: "the decision... table is a fundamental language concept... broadly applicable to many classes of information processing and decision making problems... tables force a step-by-step analysis of the decision... are easily understood by humans regardless of their functional background (they are) simple and straightforward (enough) that... specialists can write tables... with very little training... tables are easy to maintain (and) errors are reported at the source language level."

Mr. O. Y. Evans states of his work on tabular techniques: "The tabular approach... aids... in visualizing the numerous relationships and alternatives... (and) permits data rules to be readily reviewed for omissions and inconsistencies... (in addition it) provides flexibility in changing any portion of the analysis."

The CODASYL Systems Group, part of The Development Committee of the Conference on Data System Languages, has been looking into the use of decision tables. In a recent release the following statement was made: "Investigation... indicates that the systems analysis method discussed above (decision tables) will provide a precise and orderly method of documenting the analysis independent of the processing method. It will offer the analyst an aid in visualizing the relationships and alternatives of the problem, will provide flexibility in changing any portion of the analysis, and will establish a framework for the complete definition of the systems problem. The CODASYL Systems Group will continue to develop and experiment with these concepts."

To further indicate the potential results from use of tabular form, the following statements paraphrase various user opinions: Clarity and conciseness -- Decision tables are easy to prepare, read, and teach to others; experience shows that non-programmers can learn to prepare satisfactory tables in less than a day. The amount of writing, or number of words, lines and symbols used in describing complex decisions, is reduced by 25-50% as compared to flow charting. For certain specific cases, problem statement and programming time combined have been reduced significantly.

Meaningful Relationships -- Table structure serves to improve systems logic by aligning alternatives side by side. It also sharpens cause and effect understanding, so relationships which are accidental or incidental become clearer. Furthermore, actions based on similar or related conditions are apt to be drawn into the same table, making it easier to appreciate and consider interdependent factors.

Completeness -- Tabular form allows effective visual or deck debugging both by the analyst and the reviewer. There are fewer errors to start with since the analyst tends to catch his own mistakes; moreover, the reviewer will typically detect a high percentage of the remaining errors by visual examination. Finally, experience shows that with this foundation and suitable test problem construction, it is easy to rapidly detect the balance of the errors during machine debugging.

The evidence quoted on the advantages of decision tables for systems analysis and computer programming is based on actual study projects. Some of these studies even tested decision tables on various data processing machines. There are many current studies which are experimenting with a variety of tabular forms.

A Plan for Action

With all its potential advantages, it is apparent that tabular form has not yet achieved full growth and stature; there are major technical and application areas still unprobed, awaiting only the touch of creativity to make practical breakthroughs. While current table methodology does not yet provide a drawbridge to cross the communications moat surrounding systems engineers, it appears to offer the greatest chance for a significant advance.

To bring these possibilities to fruition requires experimental development. Tabular form will have to be tried and used on a wide variety of applications to provide practical evaluation and determine desirable characteristics. Along with this field pre-testing, there will be a need for effective technical developments to explore new table concepts and structures.

There are many areas which need experimental and technical development:

1. Table structure
 - multiple successes per table
 - interspersing conditions and actions
 - explicit control of sequence of actions
2. Relations among tables
 - prior rule concepts
 - use of library functions
 - use of open and closed subroutines
3. Language considerations
 - statement construction
 - macro or jargon operators
 - machine independence
4. Associated data description
 - defining factors and expressions for man-to-man and man-to-machine use
 - conditioned definitions
 - input/output format
 - preassigned values and constants
5. Implementation considerations
 - compiling vs. interpreting
 - sequential vs. random access to tables
 - possibility of made-to-order processors
 - ability to introduce specialized operators and table structures

The explosive innovations in computer hardware have not been matched by corresponding developments in systems communication. But we are on the threshold of a major breakthrough, we are on the verge of a significant advance. It's up to you and it's up to us to show equal creativity in software to that shown in hardware: To use tabular form to develop a clear, concise, meaningful, comprehensive Systems Engineering language.

T. J. Watson Research Center
Yorktown Heights, New York
August 21, 1961

Subject: Tabular Techniques Distribution #4, and
Confidential Nature of Releases

We hope you have found releases from the Systems Engineering Services Clearinghouse to be informative. These reports cover new developments in computer "software"; Tabular Techniques, one such development, shows promise of aiding our customers in evolving new applications for IBM equipment.

Many consider tables to be useful in performing the analysis necessary in designing a new system; some maintain that they are more powerful as a programming tool; others claim that the greatest benefit is derived when used as a standard documentation technique. But regardless of the area of use, we feel that IBMers need to be kept posted. For these reasons we have gathered reports of work done by customers, consultants, competitors, and IBM. Often the information contained in these papers is strictly proprietary. Two such reports were contained in Distribution #3: "Information Processing System Analysis" by Sutherland Company and "An Insurance File Maintenance Problem" by B. Grad. It is incumbent on the recipient to ensure that these reports are not duplicated or shown to non-IBMers without approval of the Clearinghouse. Your cooperation in this regard is absolutely essential.

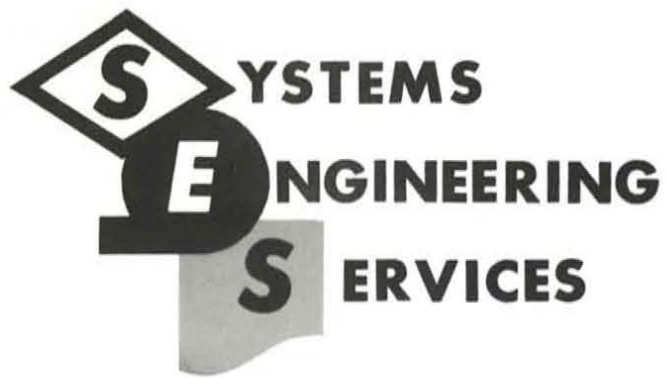
Enclosed in this distribution are two new items:

1. An article by B. Grad, entitled "Tabular Form in Decision Logic", reprinted from Datamation magazine, July, 1961.
2. A manual entitled "GE 225 TABSOL Application Manual (Introduction to TABSOL)" by the GE Computer Dept., Phoenix, Arizona.



Burton Grad, Manager
Systems Engineering Services

BG:eh
encl



CLEARINGHOUSE REPORT

TABULAR FORM IN
DECISION LOGIC

July 1, 1961
Ref. No. 1G1

Burton Grad

INTERNATIONAL BUSINESS MACHINES CORPORATION
White Plains, New York

TABULAR FORM IN DECISION LOGIC

by BURTON GRAD, IBM Corporation,
Thomas J. Watson Research Center,
Yorktown Heights, N.Y.

Reprinted from DATAMATION Magazine, July 1961

An F. D. Thompson Publication

application to computers

Since the early days of computer development, programmers have used analytical tables to convert arguments into precise functional values; they have also employed matrix structure and notation to handle common information with relatively complex structure. In the past few years, however, there has been substantial interest in probing the potential applications of tabular form for recording the decision logic itself. This exploratory work in developing decision tables has involved consideration of man-to-machine as well as man-to-man communication.

In systems analysis and computer programming, decision tables, like conventional data tables, retain a two-dimensional structure to portray significant relationships. The form, however, is considerably more elaborate to show multiple conditions and actions interlocked through position. Within a decision table any language from a business jargon to the most machine-oriented may be utilized to express the decision logic.

There are other well-known methods to describe a business system: narrative, flow charts, and logical equations. Narrative form, unfortunately, is often wordy, requiring prepositions, conjunctions, and other superfluous elements for readability; there is a certain lack of form and physical relation which may lead to inaccuracy and inconsistency if the user is not extremely careful. Flow charts require lines and connectors to show relationships; when these become too numerous, the logic may be difficult to follow and the layout may demand excessive space. Logical equations are symbolic and abstract as, for example, Boolean algebra applied to computer programming. The main limitations are the need for special skills and background to algebraically describe decision rules and the attendant difficulty in communicating equations in a business environment. Shortcomings in these well-known methods have encouraged systems analysts to take a harder look at other alternatives.

Tabular form for decision logic seems likely to satisfy this search since it compensates for many of the limitations of the other forms by providing compact expression of decision rules, visually effective display of meaningful relationships, and straightforward indication of logical correspondence. The significant difference between tabular form and other

methods is not in the notational scheme used, but rather in the physical layout for recording the systems description or programs.

Let's now examine the use of decision tables. It is not intended to suggest that this form is superior to existing languages where they are appropriate for a specialized class of problems, e.g., FORTRAN for algebraic calculations, report generators for preparing output documents. Rather, the feeling is that no method today is well-designed for systems men to use for describing complex logical decisions; therefore, decision tables may well fill a current void in a total systems analysis and programming package.

extended entry tables

One type of decision table is called EXTENDED ENTRY. Figure 2 illustrates a simple application:

	Rule 1	Rule 2		Rule 30
Age	<u>≥ 25</u>	<u>≥ 25</u>		<u>≥ 65</u>
	< 35	< 35		
Health	Excellent	Excellent		Poor
Section of Country	East	West		West
Rate/1000	1.57	1.72		5.92
Policy Limit	200,000	200,000		20,000

Figure 2

The first decision rule (columns 1 and 2) can be paraphrased: If age is greater than or equal to 25 and less than 35, and health is excellent, and section of country is East, then rate per thousand is 1.57 and policy limit is 200,000. The underlined words are implied by the table layout. The other rules are alternatives to this one, so that logically, it does not matter which rule is examined first; only one rule can be satisfied in a single pass through this decision table.

As in most disciplines, a vocabulary is needed to describe the special properties and characteristics of decision tables. Fortunately, a glossary of terms for tabular form is already

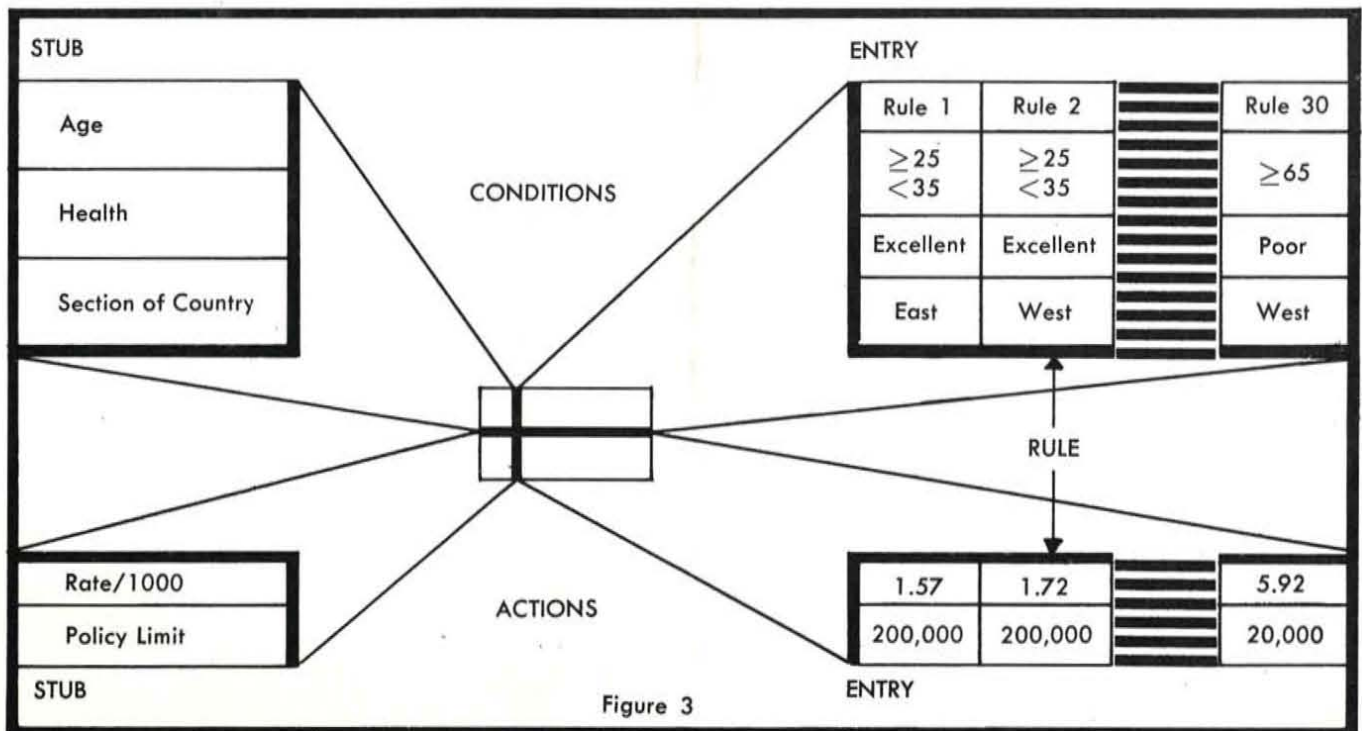


Figure 3

in existence from the statistical and financial fields; these supply an appropriate starting point.

Using the information from the insurance example (Figure 3), the decision table is shown in an exploded view, Figure 3 to show recommended titles: (see preceding page).

The double lines serve as demarcation: CONDITIONS are shown above the horizontal double line, ACTIONS below; the STUB is to the left of the vertical double line, ENTRIES are to the right. Each vertical combination of conditions and actions is called a RULE. By adding to the elements shown a title section at the top of the table which is called a TABLE HEADER, and a RULE HEADER over the entries, the essential nomenclature is complete.

limited entry tables

LIMITED ENTRY tables offer a different approach to stating the decision logic. This type of table is shown in Figure 4:

	Credit Limit is OK	Pay Experience is Favorable	Special Clearance is Obtained	Approve Order	Return Order to Sales
Rule 1	Y			Y	
Rule 2	N	Y		Y	
Rule 3	N	N	Y	Y	
Rule 4	N	N	N		Y

Figure 4

The first rule (rows 1 and 2) is read: If credit limit is OK then approve order. Again, the underlined words are implied by the form. In limited entry tables the entire condition or action must be written in the stub; the entry is "limited" to reversing a condition or ignoring a condition or action. In contrast, extended entry tables have a part of the condition or action "extended" directly into the entry. While this decision table (Figure 4) is arranged quite differently, the same table elements are present. Structurally, the table appears as in Figure 5:

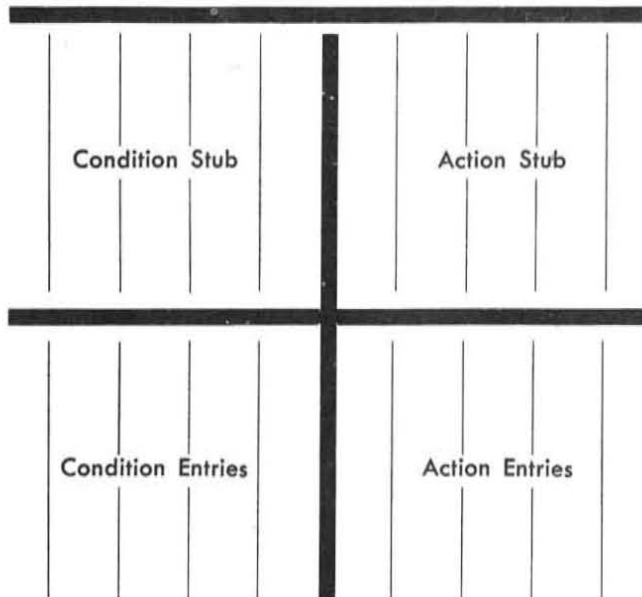


Figure 5

Limited entry permits only a few values in an entry:

Y = yes

N = no

Blank = not pertinent (e.g., condition or action need not be considered in the current rule)

business applications

Examples of successful applications of decision tables in business are as yet few in number, but some of the pioneering work can be reviewed briefly.

Initial work on the use of tabular form for recording decision logic was performed by General Electric's Integrated Systems Project from the fall of 1957 through 1959; during that period, I was the project leader. Many individuals were involved in this development work which concentrated on the use of tabular form to express the logic of product design, operation planning, cost determination, quality assurance planning, etc. This project developed extended entry decision tables for man-to-machine communication.

Mr. T. F. Kavanagh, in commenting on this work at the 1960 Eastern Joint Computer Conference,⁽¹⁾ noted, "the decision . . . table is a fundamental language concept . . . broadly applicable to many classes of information processing and decision making problems; . . . tables force a step-by-step analysis of the decision, . . . are easily understood by humans regardless of their functional background . . . (they are) simple and straightforward (enough) that . . . specialists can write tables . . . with very little training; . . . tables are easy to maintain (and) errors are reported at the source language level."

From late 1958 to the present time, Sutherland Company, a consulting firm in Peoria, Illinois, has been using tabular form for expressing what they call management decision rules. They have applied these techniques to a number of their clients' problems (e.g., a logistics study for Norton Air Force Base) with quite satisfactory results. In particular, they have used decision tables to record the logic for payroll, order processing, sales analysis, general ledger accounts, accounts payable, accounts receivable, and cost accounting. There has been no published material to date on the Sutherland work but available information indicates that limited entry decision tables are being used.

In 1959, Hunt Foods and Industries began experimenting with tabular form for man-to-man communication in computer systems planning. Material on this approach was the first to be released, in late 1959, describing how limited entry tables were used for systems analysis. Explorations were also carried out on complex relationships among individual decision using prior rule and sub-routine techniques. Many business systems were documented with decision tables: stock-control, credit analysis, sales analysis, and traffic.

In his report on the work at Hunt Foods, Mr. O. Y. Evans states, "The tabular approach . . . aids . . . in visualizing the numerous relationships and alternatives . . . (and) permits data rules to be readily reviewed for omissions and inconsistencies; . . . (in addition it) provides flexibility in changing any portion of the analysis."

Since early 1960, IBM has been actively engaged in exploring the value of tabular form both for systems analysis and for computer programming. The company has initiated joint projects with several customers to evaluate the effectiveness of various tabular forms, to explore alternative methods of implementation, and to investigate opportunities for incorporating these developments as an adjunct to existing languages. Since there are many different aspects of tabular form which still need to be examined, language implementing programs have not been prepared. These studies have developed and formalized mixed limited and extended entry tables, stubless tables, and unconditional decision tables.

The CODASYL Systems Group, which is part of the Development Committee of the Conference on Data Systems Languages, has been looking into the application and use of decision tables since late 1959. Their particular goal has been the creation of a systems-oriented language which would enable systems analysts to communicate their basic

decision logic either to computer programmers or to automatic program compilers. This organization contends that tabular form is one currently known technique which would aid in achieving effective mutual understanding of business decisions while maintaining machine independence. Their efforts have included research on generalizing tabular form to combine limited and extended entry format in a given table, as well as studies on more complex methods of sequence control, rule structure, and rule execution logic.

an example

To illustrate some of the possible advantages of decision tables, a composite tabular form is shown in Figure 6; these tables describe the logic of a file maintenance procedure. There are two input files (Detail and Master), each sequenced by identification number. The principal output is a similarly sequenced Master file incorporating additions and changes and omitting deleted records. The logic is based on having three internal areas: (1) Detail, (2) Master, and (3) New Master. "Read" as used here means "obtain the next record in the referenced file." "Write" means "produce an output Master record from the indicated source area." These are not detailed, precise tables for machine compilation, but rather the equivalent of a block diagram.

value of decision tables

So far, decision tables have been discussed in the light of known applications and attributed values and advantages.

Though many current developments are still in the realm of "company confidential," several projects have indicated results that enable us to discuss the value of tables in concrete terms.

Recalling the three benefits mentioned previously, some studies claim that decision tables appear to be superior to other methods for representing complex decision logic in that they provide or encourage:

- clarity and conciseness
- completeness
- meaningful relationships

To indicate the potential results from use of tabular form, the following statements paraphrase various user opinions: **Clarity and conciseness**—Decision tables are easy to prepare, read, and teach to others; experience shows that non-programmers can learn to prepare satisfactory tables in less than a day. The amount of writing, or number of words, lines, and symbols used in describing complex decisions, is reduced by 25-50% as compared to flow charting. For certain specific cases, problem statement and programming time combined have been reduced significantly. **Completeness**—Tabular form allows effective visual or desk debugging both by the analyst and the reviewer. There are fewer errors to start with since the analyst tends to catch his own mistakes; moreover, the reviewer will typically detect a high percentage of the remaining errors

by visual examination. Finally, experience shows that with this foundation and suitable test problem construction, it is easy to rapidly detect the balance of the errors during machine debugging.

Meaningful relationships—Table structure serves to improve systems logic by aligning alternatives side by side. It also sharpens cause and effect understanding, so relationships which are accidental or incidental become clearer. Furthermore, actions based on similar or related conditions are apt to be drawn into the same table, making it easier to appreciate and consider dependent factors.

The evidence quoted on the advantages of decision tables for systems analysis and computer programming is based on actual study projects. Some of these studies even tested decision tables on various data processing machines. There are many current studies which are experimenting with a variety of tabular forms.

future direction

With all its potential advantages, it is apparent that tabular form has not yet achieved full growth and stature; there are major technical and application areas still unprobed, awaiting only the touch of creativity to make practical breakthroughs. Current table methodology, for example, does not yet provide an effective systems-oriented language. Unable, then, to describe the decision logic in a systems-oriented language and untrained to an adequate degree in knowledge of equipment capabilities, the systems analyst often severely constrains the computer programmer.

What then of the future? Would it be desirable to directly incorporate tabular form into existing language processors such as Autocoder, FORTRAN, Commercial Translator, or COBOL, to describe complex decision procedures with decision tables? Would this approach significantly improve logical analysis? Would it simplify programming, debugging, and maintenance?

Would it be advantageous to try to create a systems-oriented language using tabular form as a primary method for describing decision logic? Should we carefully consider the relative advantages of using interpretive rather than compiler techniques for applying tabular systems-oriented languages to computers?

We are witnessing a literal explosion in scientific technology, not the least of which is the rate of innovation in computer hardware. Laboratory shop-talk treats subjects like thin magnetic films, microminiaturization, and masers, as if they were accomplished facts; and before we realize it, they often are. Progress in language concepts, though, lags seriously behind hardware advances. Failure to keep pace can be attributed to several factors: inadequate effort, requirements for compatibility with existing systems, and lack of problem recognition. Facing opportunities like automated product engineering and real-time control, we are handicapped by the limitations of current ways to describe business systems. Tabular form, one significant new tool for methods and systems people, may help to accelerate business language development and to advance systems technology.

BIBLIOGRAPHY

- (1) Kavanagh, Thomas F., "TABSOL—A Fundamental Concept for Systems Oriented Languages," Proceedings of the 1960 Eastern Joint Computer Conference.
- (2) Evans, Orren Y., "Advanced Analysis Method for Integrated Electronic Data Processing," IBM General Information Manual, #F20-8047.

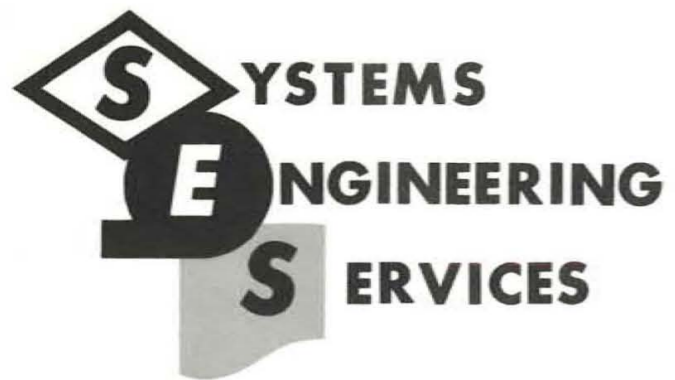
TABLE 001 — Update

Rule No.	01	02	03	04	05	06	07	08
Start	Y	N	N	N	N	N	N	ELSE
End of Detail		N	N	N	Y	Y	N	
End of Master		N	N	Y	N	Y	N	
Detail		<Master	=Master				>Master	
Detail an "Addition"		Y		Y				
Do Error Routine								X
Move Master to New Master			X					
Move Detail to New Master		X		X				
Set Addition Switch	OFF	ON	OFF	ON	OFF	OFF	OFF	OFF
Write Master					X		X	
Read Master	X				X		X	
Read Detail	X	X		X				X
GO TO TABLE	001	002	002	002	001	END	001	001

TABLE 002 — Change

Rule No.	01	02	03	04	05	06	07
Detail	<New Master	>New Master	>New Master	=New Master	=New Master	=New Master	ELSE
Addition Switch ON		Y	N		Y	N	
Detail a "Change"				Y			
Detail a "Delete"					Y	Y	
Write New Master		X	X				
Do Error Routine	X						X
Do Change Routine				X			
Do Delete Routine					X	X	
Read Master			X			X	
Read Detail	X			X	X	X	X
GO TO TABLE	002	001	001	002	001	001	002

Figure 6



CLEARINGHOUSE REPORT

GE TABSOL
APPLICATION MANUAL

July 15, 1961
Ref. No. 1G2

GE Computer Dept.

INTERNATIONAL BUSINESS MACHINES CORPORATION

White Plains, New York

This material is distributed to keep IBM personnel informed of new developments. Selection is based on interest; this department makes no claim for the desirability of this approach nor necessarily recommends its use.

If additional copies are desired, please contact the Clearinghouse. No part of this material should be reproduced or distributed outside IBM without approval of the Clearinghouse.

GE 225

TABSOL APPLICATION MANUAL

(INTRODUCTION TO TABSOL)

GENERAL ELECTRIC
COMPUTER DEPARTMENT
PHOENIX, ARIZONA

GENERAL  ELECTRIC

ACKNOWLEDGEMENT

The bulk of the material presented in this manual is based on information contained in the various publications of the Integrated Systems Project and the Services of the General Electric Company. Particular credit is due to Messers. T. F. Kavanagh, E. F. LaChance, D. F. Langenwaller, S. A. MacMullen, H. W. Nidenberg, D. T. Schmidt, and T. N. Wilcox, all of the General Electric Company, whose efforts and reports on the subject were utilized extensively in the preparation of this document.

General Electric Company
Computer Department
Phoenix, Arizona

TABLE OF CONTENTS

I. INTRODUCTION	1
II. DEVELOPMENT OF TABSOL	3
III. HOW TO READ STRUCTURE TABLES	5
IV. TABSOL APPLICATIONS	7
A. Manufacturing	7
1. Manufacturing Planning	7
2. Quality Control	10
B. Design Engineering	12
C. Finance	15
V. TABSOL & GECOM	19
VI. CONCLUSION	23

The purpose of the TABSOL Application Manual is to impart a basic knowledge of the concept and applications of TABSOL and to make present and potential customers of the General Electric Company aware of the scope and range of this new language.

No previous knowledge of TABSOL is required and a limited knowledge of computer operations is sufficient to obtain full benefit from the use of this material. The Computer Department reserves the right to make changes in the language specifications for purposes of providing the latest computer techniques to its customers.

I. INTRODUCTION

Perhaps many of you have heard the word TABSOL and have wondered "Just what is this concept that everyone is taking about." It is to those of you who have never heard of this term before, that this publication is directed.

The objective is to remove the aura of mystery from the subject and present in a clear, concise manner the history, development, and potential use of this new language in the industrial world. Illustrations of potential applications of the TABSOL language in the areas of Manufacturing, Engineering, and Finance are described in detail and the tremendous power that GECOM (General Compiler for GE Computers) lends to TABSOL is demonstrated.

TABSOL, which stands for Tabular Systems Oriented Language, is basically a structuring technique used to systematically describe the step by step decision logic in the process of solving a problem. The basic advantage of the TABSOL language is that it is probably one of the most easily learned and understood and can be applied to many analytical situations.

The tabular technique is not new to industry. Tables have been used for sometime as an aid in problem solution. When the manufacturing planner sets up a price table for the planning of coil forming he uses a tabular technique. When the air conditioning design

engineer refers to the refrigerant pressure vs. temperature table he is also using the tabular technique to aid in solving the problem. Tables are designed to aid the user in determining specific relational characteristics.

The TABSOL structuring technique involves the use of a table to facilitate the function of specifying decision logic. Computer programming is a perfect example of the job performance that can be improved with the application of this method. The computer programmer receives functional specifications and decision logic from the systems analyst and, in turn, translates this logic into a language that a computer understands. When the programmer speaks to an engineering analyst he must converse in engineering terms. When involved with an accounting analyst a different language is used. The translation of these terms for computer usage generally involves displaying the system logic by means of a flow chart from which the program is written.

TABSOL v 225 which is the union of TABSOL with GECOM enables the advantages of tabular structured decision logic to be supplemented with all the power of the most up to date compiler ever written. This marriage permits the systems analysts to prepare all inclusive decision tables for direct input to General Electric Computers, significantly reducing programming time and effort.

II. DEVELOPMENT OF TABSOL

A General Electric Company task force, formed in 1957, developed a system which converts customer orders into finished products automatically. The system covers order editing, engineering design, manufacturing operation planning, product cost determination and manufacturing control. In developing an automatic system with the many inherent complexities it was apparent that some means of reducing programming and coding effort was required. The structure table was developed to satisfy this requirement and defines the precise manner in which information must be written in order that all elements of the logical decision are in the proper position.

The solution of these structure tables in a computer is simplified by the use of TABSOL, a generalized, automatic method by which a computer can solve any structure table regardless of content. The Integrated Systems Team used this feature to carry information through from the customer's order to shipment of the finished product.

The first efforts of the General Electric task force were directed toward writing interpretive type TABSOL programs. These programs were first used at the Company's Instrument Department and the team had achieved a major breakthrough in automatic language development. However, from that point on until the development of TABSOL 225 there still existed the serious limitation that despite the effort of the design engineer, manufacturing specialist, and others, in constructing their decision logic in tabular form it was still necessary to expend considerable effort in a detailed coding operation to put the tables in a language the computer could understand.

But progress was being made and, despite this obstacle, the concept of structuring itself offered such potential that a great degree of interest was generated within General Electric Company. Other components of the Company, with the aid of the interested service organizations began to explore the possibilities in their own fields and with their own machines.

TABSOL was applied to design engineering problems, manufacturing planning and quality control problems, and financial and cost control problems. The enthusiasm that was generated began to multiply. In all cases the language was a powerful tool towards the development of an integrated mechanized system with the resulting cost savings.

During the rapid growth in the development of the concept, there still remained the problem of the detailed coding requirements. To be sure, the techniques were improved to such an extent that anyone could do the coding with little knowledge of the content of the table.

However, in late 1960 the General Electric Company made two announcements of great significance. The first was the formal announcement to the public of TABSOL - A Fundamental Concept For Systems Oriented Language by T. F. Kavanagh, who was instrumental in the development of the tabular concept, at the Eastern Joint Computer Conference in New York. The second announcement was by the General Electric Company's Computer Department concerning the General Compiler (GECOM) for GE machines. Part of the release stated "The Computer Dept. now offers with the GE 225 the Tabular Systems Oriented Language (TABSOL 225), the first "Systems Oriented" language to be processed by a compiler".

This was the breakthrough for which the early table user's were waiting. It meant that the power of a full fledged language was at the command of every structure table entry. With this automatic program, it was now possible to feed decision tables, as prepared by the analyst, directly to the General Compiler for processing. The program produced by the compiler is tailored according to the analyst's specifications and the GE 225's capabilities. Thus, a new language that can be used by itself or in conjunction with all the features available in GECOM, puts control of the electronic computer within the reach of additional scores of engineers, scientists and systems analysts.

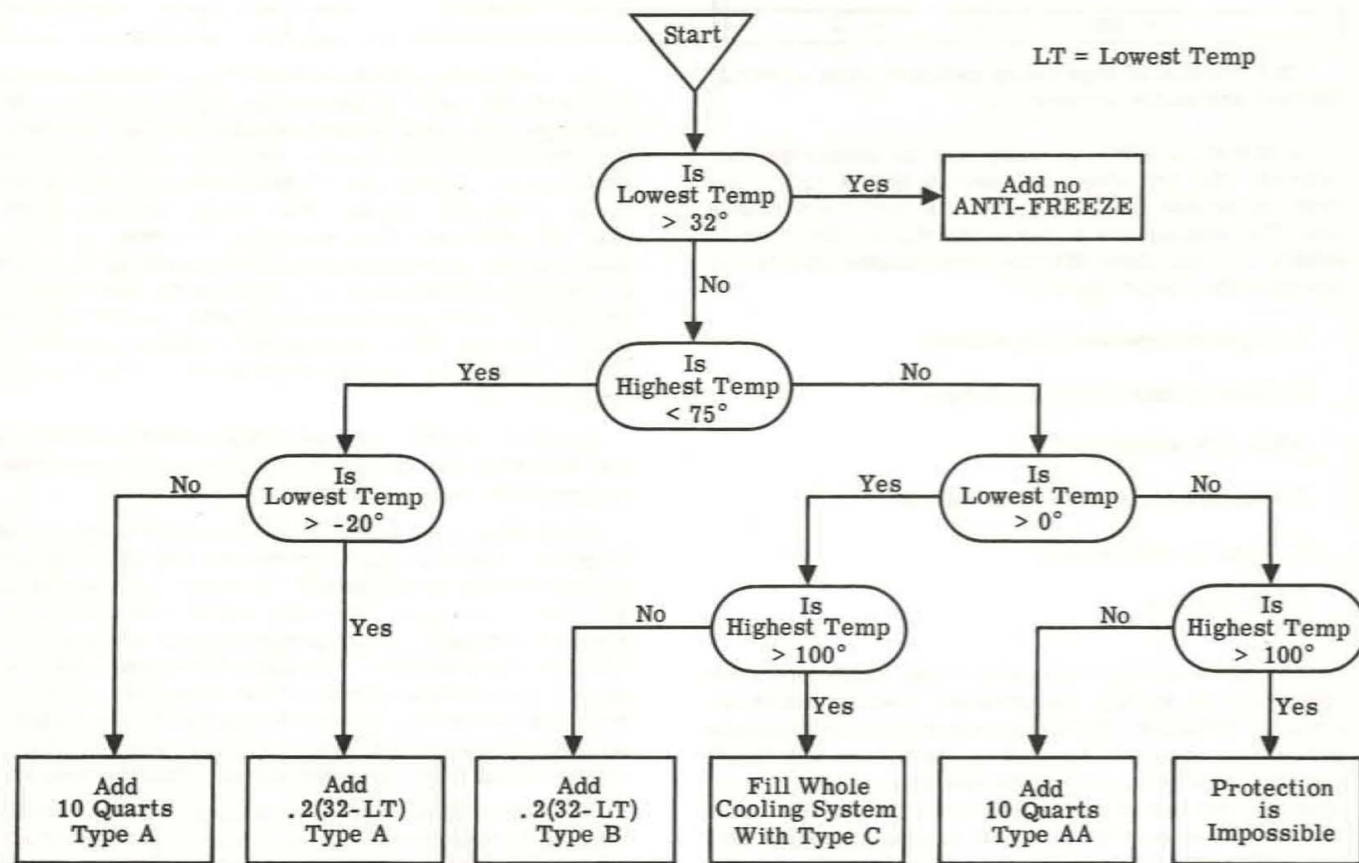
III. HOW TO READ STRUCTURE TABLES

In order to demonstrate the use of the tabular concept and the method by which it is interpreted, let us consider an illustration. Consider the problem of a foreign car manufacturer who must add anti-freeze to the cooling system of his car in varying amounts depending on the delivery point of the automobile. Of course, the amount and type of anti-freeze depends on the value of two controlling factors - these are the highest and lowest temperatures to which it is expected the car will be exposed. The decision pattern that he uses is as follows:

- (1) If the temperature is greater than 32°F add no anti-freeze.
- (2) If the temperature range is from -20°F and below to less than 75°F, add 10 quarts of type A anti-freeze.
- (3) If the range is from above -20°F to less than 75°F, add .2 (32 - lowest temp) quarts of type A anti-freeze.

- (4) If the range is from 0°F and below to 100°F, add 10 quarts of type AA.
- (5) If the range is from above 0°F to 100°F, add .2 (32 - lowest temp) quarts of type B anti-freeze.
- (6) If the range is from above 0°F to above 100°F, fill the whole cooling system with type C anti-freeze.
- (7) If the range is from 0°F and below to above 100°F, then protection is impossible.

If a computer programmer were given this problem, his first step would be to set up a flow chart which would depict the steps required for the computer to proceed through the decision making process. His flow chart might look like this:



Although the flow chart is a clear, concise statement of the problem to its original author, it could present a serious problem in interpretation to anyone who attempted to use it as a basis for giving a computer detailed instructions for its solution.

As a matter of fact, one of the most serious problems existing in the programming field is that of communication between programmers on problems already solved. It is a widely held axiom that it may be better to re-write a flow chart and program rather than to try to interpret those written by someone else. This problem is considerably reduced with the use of TABSOL.

Let us express the same process in the form of a structure table.

Highest Expected Temperature	Lowest Expected Temperature	Amount of Anti-Freeze in quarts	Type of Anti-Freeze	Go to Table
-	> 32	0	-	6
< 75	≤ -20	10	A	6
< 75	> -20	.2 (32-LT)	A	6
≤ 100	≤ 0	10	AA	6
≤ 100	> 0	.2 (32-LT)	B	6
> 100	> 0	Capacity of cooling system	C	6
> 100	≤ 0	-	-	6

This method of expressing decision logic is easily learned and easily understood.

A structure table is composed of conditions and actions. The conditions are stated to the left of the vertical double line and above the horizontal double line. The actions are stated to the right of the vertical double line and above the horizontal double line. In our example the conditions are:

1. Highest expected temperature
2. Lowest expected temperature

While the actions are:

1. Amount of anti-freeze in quarts
2. Type of anti-freeze
3. Go to table

The table is then composed of any number of rows necessary to specify the possible alternatives of the problem situation. Each row is evaluated in sequence proceeding from the top row to the bottom row. If all the conditions of a row are satisfied then all the corresponding actions in that row are executed and the table is considered solved. It is, of course, possible that the conditions in a number of rows are satisfied at the

same time. For example, if the temperature range in a particular location were from 10° above zero to 70° above, then the conditions for rows 3 and 5 are both satisfied. However, since we proceed row by row until the conditions are satisfied, we can obtain only one solution to any table - in this case, row 3. This particular point illustrates the care that is necessary in constructing tables so that they actually represent the problem to be solved. This care, of course, is not required if the table can be constructed with row independence, that is, where one and only one row can be a solution to the problem. When tables are constructed with row independence, then those rows that are most likely to be solution rows should be placed at the top of the table offering potential speed advantages. Of course the systems analyst must weigh the alternatives when constructing the tables.

It is common practice in reading structure tables to insert the word if before the stated condition, the word and for each vertical single line, and the word then for the vertical double line. If any particular condition is not significant to the solution it may be left blank in the table or the letters N.S. (not significant) may be inserted. For example, the reading of the third row of our example would be something like this. If the highest expected temperature is less than 75° and if the lowest expected temperature is greater than minus 20, then the amount of anti-freeze in quarts is .2(32-lowest temp) and the type of anti-freeze is A and go to table 6.

Again, it should be noted that if either condition is not satisfied, the program proceeds to the next rows successively until a solution row is obtained.

This, then, is a basic illustration of the concept of TABSOL, and all tables written in the language are interpreted in this manner. Of course, in a typical manufacturing application the number of tables required for solution of the problem could easily exceed 100 and the numbers of actions, conditions and rows would vary up to the limits of the computer. The only requirement of the system is that the entire problem be clearly thought through so that all elements affecting the solution are considered in the tabular formation.

With this knowledge of the tabular concept and the method of interpretation, we can now proceed to examine the potential that it offers to computer users.

IV. TABSOL APPLICATIONS

As with any new industrial development, it is necessary to educate potential users in the application of the new tool so that it may be applied to their operation. Since the development of TABSOL has occurred over the past 5 years with a major breakthrough occurring just months ago, there is a wealth of information to be disseminated. The purpose of this presentation is to show some of the typical applications of TABSOL and the potential that these applications offer. The applications described are, of course, only illustrative such that no conclusions about the actual data used should be drawn or questioned. The only objective is to demonstrate the potential uses of TABSOL.

A. Manufacturing

The Manufacturing Section of a business normally consists of the following operations:

Materials - Procurement, Scheduling, Dispatching, Inventory Control

Manufacturing Engineering - Operations Planning, Machine Development

Quality Control - Appraisal, Testing

Shop Operations - Manufacture and Assembly of Parts and Components

Administrative - Personnel, Budgets, Systems.

It would be impractical to give a detailed description of a TABSOL application for each of these areas. However, there are two typical applications which would serve our purpose. In these illustrations enough detail is given to provide a clear picture of the actual system but not so much detail that the picture becomes confusing.

A.1 TABSOL in Manufacturing Planning

A typical application is a system that provides complete manufacturing operation planning for the assembly of cast rotors.

Consider the planning operation in a motor manufacturing concern. The specific operation to be planned is to stack and to press a specific rotor.

The planner, in issuing detailed instructions to the factory may issue a pre-printed planning form that looks something like that shown in Figure 1.

GENERAL ELECTRIC COMPANY					
OPERATION PLANNING SHEET					
Drawing Number		Shop Order Number		Quantity	Schedule Date
Oper. Number	Operation Description	Work Station	Operation Time (Minutes)	Set-up Time (Minutes)	Total Price
1	Get ① O.D. ② High Arbor		⑧	⑨	⑩
	Get ③ I.D. ④ O.D. ⑤ Thick Collar				
2	Stack ⑥ Stacks ⑦ inches				

Figure 1

In order to fill in the form he must determine the following information: (The numbers correspond to those in the blanks on Figure 1)

- (1) Lamination OD
- (2) Arbor Height
- (3) Lamination ID
- (4) Collar OD
- (5) Collar Thickness
- (6) Number of Stacks
- (7) Stack Height
- (8) Operation Time
- (9) Set up Time
- (10) Operation Price

Regardless of what model motor must be planned, the planner must make a determination of each of the above quantities.

Assume that planning is required for a motor with the following characteristics (for the rotor):

- 2 pole rotor
- Stack height 12"
- Lamination OD of 10"

The planner, with this information, plus his own "planning lore" can now fill in the blanks of the operation card.

His own logical thinking process flows in the following pattern:

Planner Thinking:

- a. "It's a two pole motor, therefore the lamination ID is 5.0 inches" (he writes lamination ID equals 5.0 inches)
- b. "The stack height is 12 inches; therefore the stack height - arbor height conversion table says the arbor height is equal to 17.0 inches" (he writes arbor height equals 17 inches)
- c. "Collar thickness equals arbor height minus stack height; equals 5 inches" (he writes collar thickness equals 5 inches)
- d. "Do we have a collar with that thickness available?" (Checks list and finds that collars are available from 1 to 5 inches in 1/8 inch increments) "There is a 5 inch collar available."

- e. "If the lamination OD is less than 15.00 inches, the collar OD is 7.00 inches." (he writes down collar OD of 7.00 inches)
- f. "The stack height - stack quantity table shows that only 1 stack is required." (he writes down number of stacks equal to 1) He then calculates the stack and press operation time:

For 12 inch stack height the stack and press operation time equals 110 minutes + 5 minutes for each inch of stack height = 110 + 60 = 170 minutes and the set up time = 25 minutes.

The price for this is (looking up the table for prices on stack and press operations) \$.02 /minute plus .015/minute set up. Therefore the price is \$3.775 for this operation.

At this point the planner has gone through all the gyrations necessary to obtain the required information for the planning card. After a few passes through something as simple as this the planner becomes quite proficient at preparing planning sheets such as these. However, the routine is quite repetitive with but minor changes in the various measurements. If some way were devised such that these figures could be inserted for any rotor we could easily computerize this decision routine. It would provide the further advantage of being universal in that all rotors of all motors could be passed through the system and data could be generated automatically.

Consider the same logic pattern in tabular form.

Recall that inputs to the system are number of poles, stack height and lamination OD.

No. of Poles	Lamination ID	Go to Table
= 2	5 inches	2
> 2	N. S.	10

TABLE 1

This table is read "If the number of poles equals 2 then the lamination ID is 5 inches and go to table 2". If the table did not solve in the first row it is read "If the number of poles is greater than 2 then the lamination ID is not significant (to this table) and go to table 10" which presumably leads us down the road toward finding the lamination ID for motors with more than 2 poles.

Since the rotor now in question is a 2 pole rotor, the computer remembers that the lamination ID is 5 inches. It then proceeds promptly to Table 2 to calculate the arbor height collar thickness, operation time, and set up time for the job.

Overall Stack Height	Arbor Height	Collar Thickness	Stack & Press Operation Code	Stack & Press Operation Time	Set up Time	Go to	
≥ 2	< 5	7	7 - Stack Height	1	100 + 5 (Stack Height)	20 Min	Table 3
≥ 5	< 10	12	12 - Stack Height	1	100 + 5 (Stack Height)	20 Min	Table 3
≥ 10	< 15	17	17 - Stack Height	1	100 + 5 (Stack Height)	25 Min	Table 3
≥ 15	< 20	22	22 - Stack Height	1	100 + 5 (Stack Height)	25 Min	Table 3
≥ 20	< 25	27	27 - Stack Height	1	100 + 5 (Stack Height)	25 Min	Table 3

TABLE 2

In Table 2, Row 1, the computer asks:

- (1) Is overall stack height greater than or equal to 2? Ans. Yes.
- (2) And is overall stack height less than 5? Ans. No.

Since it did not solve in the first row of the table it proceeds to the second and subsequent rows until all questions before the vertical double line are answered "yes", which in this case occurs in row 3. The computer then records in its memory that the arbor height is 17 inches, that the collar thickness is 17 minus stack height or 5 inches, the operation code is 1, the operation time is 110 minutes plus 5 minutes for each inch of stack height for a total of 170 minutes. The set up time is recorded as 25 minutes and the computer passes to TABLE 3.

Lamination OD	Collar OD	Go To
> 5	≤ 10	7.00 inches
> 10	≤ 15	10.00 inches
> 15	N.S.	13.00 inches

TABLE 3

Reading table 3 we see that the collar OD is strictly dependent upon the lamination OD and since our lamination OD is 10 inches the computer determines that the collar OD is 7.00 inches and proceeds to TABLE 4 to determine the pay rate for this operation.

In table 4 we have the rate per minute for each operation. Note that this table is used for more than the stack and press code (code 1).

Price Calculation After table 4 the computer enters this part of the program and determines that the job price equals operation time times rate per minute plus set up time times set up rate per minute.

In our example Job Price equals $170 (.02) + 25 (.015) = 3.775$. The computer after performing this calculation proceeds to the output portion of the program and generates on preprinted forms the necessary planning data. (Figure 2)

The computer has thus carried out the same routines that the planner had in a much shorter time. Through the tabular method it was able to make all required logical decisions.

Although the planning operation could have been computerized by conventional programming methods, let us examine the advantages obtained by using the structure table concept.

By structuring the problem a precise and complete documentation of the logic involved is available. Additionally this logic is broken down into several individual packages (the tables themselves) each of which can be examined for consistency. This breakdown aids in bringing errors to light and points out potential opportunities for standardization.

Another major advantage is that changes can readily be incorporated into the system promoting increased accuracy in control systems. Some present day methods of operation are so cumbersome that many changes are not incorporated because the implementation cost is more than could be justified by the improved accuracy.

Operation Code	Rate/minute	Set up Rate/min.	Go To
1	\$.02	\$.015	Price Calculation
2	.025	.015	Price Calculation
3	.03	.015	Price Calculation
4	.035	.015	Price Calculation

TABLE 4

GENERAL ELECTRIC COMPANY					
OPERATION PLANNING SHEET					
ABC Drawing Number	123 Shop Order Number		X Quantity	Feb. 30, 1961 Schedule Date	
Oper. Number	Operation Description	Work Station	Operation Time (Minutes)	Set-up Time (Minutes)	Total Price
1	Get <u>10"</u> O.D. <u>17"</u> High Arbor Get <u>5"</u> I.D. <u>7"</u> O.D. <u>5"</u> Thick Collar Stack <u>1</u> Stacks <u>12</u> inches	1	170	25	\$3.775
2					

Figure 2

The biggest advantage however, and one which can be obtained only with the use of TABSOL 225, is that the functional specialist can now write, check and update the tables for direct input to the GE 225 computer. There is no longer any communication problem between analyst and programmer.

With TABSOL 225 the planning specialist now needs only to develop the logic of the system as direct input to the manufacturing planning operation. The manipulation of numbers is transferred from the planner to the computer which performs these operations much more economically offering complete mechanization of routine planning.

The key to success, as we have seen, in these applications is a basic understanding of the logic behind decisions. It is necessary to capture and define this logic if the planning system is to make decisions without the aid of the planning specialist, on parts that were never physically produced before. The structure table represents the most efficient and easily understood method for specifying the planning decision logic.

A.2 TABSOL in Quality Control

The quality control operation of the manufacturing function is responsible for the assurance that the product being shipped to the customer conforms to all engineering specifications. It is, therefore, the group

that performs the necessary inspections, tests and reliability studies to ensure the manufacture of a quality product.

In order to perform the inspection portion of the quality control operation, the inspector must be provided with the knowledge of what to inspect, what equipment to use, how often to inspect, size of sample, etc.

The structure table provides a convenient, economical method for providing the decision logic and TABSOL 225 makes the mechanization of this logic a fairly simple process.

Consider the requirements at an in-process inspection station for bevel gears. The objective is to provide the inspector with sufficient information to completely appraise the gear.

Sufficient information may consist of:

- a) Inspection points
- b) Dimensional characteristic of each inspection point
- c) Required inspection tools
- d) Tolerances permitted
- e) Classification of characteristics

Number of Teeth	Diametral Pitch	Tooth Length	Gear O.D.	Front Angle	Back Angle	Go to
20	6	.650	3.5	51°	45°	Table 2
23	5.50	.875	4.6	50°	45°	Table 2
23	5.25	.950	4.7	49°	45°	Table 2
25	5.00	1.000	5.2	49°	45°	Table 2
25	4.50	1.500	6.1	49°	45°	Table 2
27	20	.25	1.4	48°	45°	Table 2
-	-	-	-	-	-	Table 100

TABLE 1 Main Winding Number of Turns

With this information in the hands of the inspector he will be able to perform the necessary operations to determine whether or not the product has been made to specifications.

In the bevel gear example it is necessary to inspect the tooth length, gear outside diameter, the front angle and the back angle. The actual dimension for these characteristics is dependent upon the number of teeth and the diametral pitch of the gear. Table 1 (above) can then be set up to provide the decision logic for this operation.

The first line of the table says "if the number of teeth is 20 and if the diametral pitch is 6 then the tooth length is .650 and the Gear O.D. is 3.5 and the front angle is 51 degrees and the back angle is 45 degrees and Go to Table 2". Proceeding to Table 2 the inspector is provided with the proper pinpoint micrometer size to check the tooth length of the particular gear. The tooth length was an output of the previous table.

Tooth Length		Pinpoint Micrometer Size	Go To
> 0	≤ 1	1 inch	Table 3
> 1	≤ 2	2 inch	Table 3
> 2	≤ 3	3 inch	Table 3

TABLE 2 Main Winding Wire Diameter

This type of table is generally used to provide proper equipment selection for required dimensional checks in any quality control system. Now that the inspector has been provided with the characteristics requiring measurement and the tools required to appraise that function he must also know the tolerances for each of the listed dimensions.

The tolerance of the runout on the front angle is a function of the size of the outside diameter. The larger the O.D. the greater the tolerance. The actual allowable tolerance is shown in Table 3.

Gear O.D.		Front Angle Run Out	Go to
> 0	≤ 2	.0007	Table 4
> 2	≤ 4	.0010	Table 4
> 4	≤ 6	.0011	Table 4
> 6	≤ 8	.0014	Table 4

TABLE 3 Main Winding Wire Material

Of course certain tolerances are fixed, i.e., they are constant regardless of the dimension of the particular characteristic. These tolerances can be generated for any quality control operation as shown in TABLE 4. Since the conditions in TABLE 4 are also necessary for the classification of characteristics, it can be utilized for this purpose as well. The classification of characteristics is necessary for the proper utilization of sample size tables toward attainment of the desired quality level.

Characteristic	Tolerance	Classification
Tooth Length	.01	Major
Gear O.D.	.05	Major
Front Angle	0° 8'	Major
Back Angle	1° 0'	Minor

TABLE 4 Main Winding Wire Specification

The inspector is now able to perform the appraisal job for the bevel gear. For any gear in production the computer, by means of the structure table, will be able to generate the written data required to adequately perform the appraisal function. GE TABSOL 225 makes the implementation of this type of program feasible for any quality control operation.

Other potential applications within quality control which lend themselves particularly well to the structuring technique include Process Capability Tables,

Quality Time Standards determination, Acceptable Quality Level (AQL) determination, etc. A good quality control system will include all of these operations in the process of measuring product quality.

Some of the positive benefits that quality control operations obtain with the use of structure tables are:

- a) reduction in total quality cost - by providing a rapid and regenerative means for developing quality instructions.
- b) better product quality - due to the increased ability to provide specific, accurate and pertinent instructions to the shop for each manufacturing operation.
- c) provides the quality planning and process control requirements automatically, through the use of a computer to shop operators, inspectors and testers.
- d) improves manufacturing cycles by reducing production delays due to poor quality.
- e) provides a disciplined and automatic means for integrating the quality needs of a product line between engineering and manufacturing.

SUMMARY

Thus the application of TABSOL to two primary operations within the Manufacturing function has been described. There are others, in Materials, Shop Operations, etc. which are not described here, that offer equally great opportunity for improved operations and cost savings.

Because of total system complexity the method used for organization of data must, of necessity, be versatile. The structure table technique by itself satisfies this requirement. The fact that this same system can be used as a direct input to the computer demonstrates the vast power of this new methodology.

By using the structuring technique described here the Manufacturing Systems analyst has great opportunity to reduce cost and increase the ability of the Manufacturing Section to deliver high quality products on time. The technique is such that all of manufacturing can be tied together into a smooth working unit with the decisions of each of the components falling into a flow pattern.

B. Design Engineering

Much effort and progress in the utilization of the structure table technique has occurred in the engineering function. Since engineering design information is used extensively throughout Manufacturing and Finance it is desirable that documentation that can easily be

used by these other operations be provided. The structure table thus provides a two fold benefit. For the finance man or manufacturing man we have a communication technique whereby the engineer can be easily understood. Whereas in the past it may have appeared that design decisions were made strictly at random, it is now possible to communicate the long sought after logic behind the decision. With this new-found knowledge the Manufacturing and Finance people are able to offer positive recommendations to Engineering regarding the effects of engineering decisions on their operation.

The second benefit is that the structure table enables a design engineer to see the entire scope of a component at any particular time. An entire group of structure tables can convey the data for all components of all models. Since our new form of documentation is more compact than the present drawings and parts lists, it is much easier to manipulate information in the study of particular design problems.

Consider the applications of structure tables to the design function. Most design decisions are determined by:

- a) Customer Requirements
- b) Process Capability
- c) Cost
- d) Technology

With this information known the individual design engineers begin to design the product. The problem facing the engineer at each decision step is whether or not he is using the optimum material at this point or the optimum dimensional characteristics in light of what has been designed before. Proper decisions at this point can reduce material costs, cycle time and labor costs, all of which are direct elements of manufacturing. What then is required is some means by which the design engineer can have this information available so that the best possible decision can be made. Design Structure Tables provide this flexibility as the logic behind the design decisions are recorded. When a new product of a particular product line is designed it can easily be incorporated into the present design structure. The technique of designing an entire product line at one time rather than individually tends to provide a reduction in cost because of the ability to maintain consistency between products. For example, in one case a variety of thicknesses of sheet metal had been selected to make chassis for electronic equipment. This variety included fourteen different thicknesses whereas a subsequent engineering examination revealed that three could have served just as easily. These situations arise because the design engineer making the decision in many cases does not have readily available the information necessary to determine the optimum characteristics.

Let us look at some design tables to see how structure tables apply to this and other design engineering problems. Since we are now able to read the tables with greater ease and facility we shall go into a little more depth at this stage:

Consider an Instrument Armature. The design engineer is required to specify the following items of information:

- a) Main Winding Wire Diameter
- b) Main Winding Number of Turns
- c) Main Winding Wire Material
- d) Main Winding Wire Specification
- e) Damper Winding Wire Diameter
- f) Shaft Body Length
- g) Shaft Body Diameter
- h) Shaft Body Material

The input information that the engineer receives from Marketing is typically

- a) Type of Service AC or DC
- b) Rating - in AMPS, MICROAMPS, MILLIAMPS, MILLIVOLTS, VOLTS and WATTS

With these items of information he sets out to provide the required design data. The first table is established to determine the Main Winding Wire Diameter.

Reading the first line we ask: If the type service is DC and if the rating units are microamps and if the unit rating value is greater than 180 and if the unit rating value is less than 450 then the wire diameter in mils (.001) is 1.0 and go to Table 2. Note that the last row is designed such that if none of the previous conditions were satisfied we proceed to Table 100 which will follow the procedure required for a special instrument. Table 2 is a continuation of the same process but is designed to provide the number of turns in the main winding.

All the design tables for the process of specifying the characteristics for the instrument line are reproduced here. From this set of tables it becomes obvious that consistency between models will be maintained. The reasons behind each of the decisions is clearly stated.

Again take note that if none of the conditions are completely satisfied we proceed to Table 100 for handling of specials.

Thus we see that the computer can go through the tables and give complete specifications for almost all instruments. Those that are special (not provided for in the tables) go to the design specialists who, depending upon his analysis of the situation, decides whether or not to expand the tables for their provision. The tables, therefore, are not taking any decisions away from the design engineer. Indeed, they are only a structure of the logic for those decisions the engineer has already made. The design engineer is now free to devote all his time to the design problem of specials at which point the structure table tool is also a positive aid. We have thus provided a method of operation for the design engineer which provides the advantages listed following Table 8.

Service	Rating Units	Rating Value \geq	Rating Value $<$	Wire Diameter in MILS	Go to Table
DC	μA	180	450	1.0	2
DC	μA	450	900	1.25	2
DC	MA	0.90	1.80	1.50	2
DC	MA	1.80	4.50	2.0	2
DC	MA	4.50	9.20	2.5	2
DC	MA	9.20	13.50	3.0	2
DC	AMPS	0.8	66.0	8.0	2
DC	MV	45	330	8.0	2
DC	VOLTS	0.9	300	2.0	2
DC	VOLTS	300	1100	1.5	2
AC	WATTS	---	---	2.0	2
AC	VOLTS	---	---	2.0	2
--	--	---	---	---	100

TABLE 1 Main Winding Wire Diameter

Service	Rating Units	Rating Value ≥	Rating Value <	Number of Turns	Go to Table
DC	MA	0.18	13.5	300/I	3
DC	MA	13.5	18.0	26	3
DC	MA	18.0	23.0	15	3
DC	AMPS	0.023	66.0	13	3
DC	VOLTS	0.9	300	60	3
DC	VOLTS	300	1100	120	3
DC	MV	45	150	26	3
DC	MV	150	330	13	3
AC	WATTS	---	---	230	3
AC	VOLTS	---	---	230	3
--	--	---	---	---	100

TABLE 2 Main Winding Number of Turns

Service	Rating Units	Rating Value ≥	Rating Value <	Wire Material	Go to Table
DC	MA	0.18	13.5	Copper	4
DC	AMPS	0.0135	66	Aluminum	4
DC	MV	45	330	Aluminum	4
DC	VOLTS	0.9	1100	Copper	4
AC	WATTS	---	---	Copper	4
AC	VOLTS	---	---	Copper	4

TABLE 3 Main Winding Wire Material

Service	Rating Units	Rating Value ≥	Rating Value <	Wire Specification	Go to Table
DC	MA	180	220	B50W133C	5
DC	MA	0.22	13.5	B50W133B	5
DC	AMPS	0.0135	66	B50W217	5
DC	MV	45	330	B50W217	5
DC	VOLTS	0.9	1100	B50W133B	5
AC	--	--	--	B50W133B	5

TABLE 4 Main Winding Wire Specification

Service	Rating Units	Rating Value ≥	Rating Value <	Wire Diameter in MILS	Go to Table
DC	MA	180	220	3.0	6
DC	MA	220	450	4.0	6
DC	AMPS	0.00045	66	8.0	6
DC	MV	4.5	330	8.0	6
DC	VOLTS	0.9	1100	8.0	6
AC	---	--	--	NONE	6

TABLE 5 Damper Winding Wire Diameter

Service	Rating Units	Length (Inches)	Go to Table
DC	--	2.121	7
AC	AMPS	1.979	7
AC	VOLTS	1.979	7
AC	WATTS	3.981	7

TABLE 6 Shaft Body Length

Service	Diameter	Go to Table
DC	0.0061	8
AC	0.072	8

TABLE 7 Shaft Body Diameter

Service	Material	Go to
DC	Aluminum	End Routine
AC	Bronze	End Routine

TABLE 8 Shaft Body Material

- Structure tables are easy to read and understand.
- The design logic is presented in a simple, straight forward manner.
- The tables become a useful information source.

- The table format shows the boundaries of the design and clearly points out incompleteness or inconsistency.
- The tables can be a direct input to manufacturing in an integrated system.
- Structure tables are easily solved by the GE225 computer.

C. Finance

Now that we have explored the potential of structure tables in design engineering and manufacturing we can consider expanding the system to include product costing. The major requirement for this development is that manufacturing, engineering and finance must work in a completely integrated fashion so that all necessary financial data is obtained or generated at the most logical point in the system.

To provide some understanding of the methods used to develop the cost of a product, the present method shall be described before the new method is developed.

Model lists, material lists and drawings are obtained from engineering. The Finance Section makes up a separate cost card for each part, assembly and model and enters the following data on the cost cards:

- Dimensions or weight of material
- Material Specification
- Quantity of parts
- Name of part, assembly or model

Finance then obtains from Manufacturing the operational planning cards and adds more information to the cost cards, namely:

- Time value or price of each labor operation
- Job rate
- Sequence of operations

The standard labor values and standard material values are then calculated for each part and they are summed and entered on the cost cards. These steps are repeated until the standard cost of each part, assembly and model is determined. These cost cards are used for obtaining the standard direct material and the standard direct labor values of a completed or partially completed part or assembly. These data are required in order to obtain:

- a) Cost of production
- b) Cost of Shipments
- c) Cost of Scrap
- d) Cost of inventory

It is apparent from a systems point of view that the present method is based on the file reference technique rather than on the regeneration concept.

If structure tables were used to generate product costs, the computer would go through the following steps to determine the cost of a model.

a) The parts characteristics, which are output from the engineering structure tables are input to the Finance Structure Tables. These characteristics determine the cost values that will be obtained for a particular part or assembly upon the solution of the cost structure tables.

b) After the proper cost value is obtained for the specified parts characteristics it is temporarily stored in memory until all of the cost tables have been solved. When the model costs are calculated the parts cost will thus be available.

c) A series of structure tables will then be used to build up the costs in the proper order for the particular models.

An example may be in order at this point to illustrate the types of structure tables that would be used to calculate the direct material cost and direct labor cost of a sample product.

In our example take note of the fact that the inputs to the cost structure tables are outputs from Manufacturing and Engineering.

Number of Turns		Allowed Time in Seconds	Operator Class	Go to Table
> 0	< 15	15	1	4
≥ 15	< 100	40	2	4
≥ 100		150 + √ no. of turns	2	4

TABLE 3

The first table will be one in which the cost per hundred pounds of material is determined.

Thus if the material specification is B50W70 and the wire diameter is 2 MILS then the cost per hundred pounds is \$150.00 and we go to Table 2 to determine the material weight per hundred coils. Note that Table 2 requires a knowledge of the number of turns in the coil, which is also an output from the design structure tables.

Material Specification	Wire Diameter (MILS)	Cost per C Lbs.	Go to Table
B50W70	2	\$150.00	2
B50W70	4	\$120.00	2
B50W70	6	\$100.00	2
B50W200	8	\$ 95.00	2

TABLE 1 Material Cost Table

Kind of Material	Material Weight per C Coils	Go to Table
B50W70	Turns x (Diam) ² x .000082	Cost Formula
B50W200	Turns x (Diam) ² x .000025	Cost Formula

Material Cost = Cost per Hundred Pounds x Weight per C Coils

TABLE 2

Note that the cost formula is not in tabular form as there would be no need for it in the computer program since it is the same for all materials, shapes or form.

The direct labor costs are generated in the same manner as the material costs were developed. Note that inputs to these tables are outputs from manufacturing and engineering design structures.

Operator Class	Job Rate (\$ per second)	Go to
1	.030	LABOR FORMULA
2	.040	LABOR FORMULA
3	.050	LABOR FORMULA

Labor Cost/C = Time Allowed x Cost/Second x 100

TABLE 4

These tables serve as an illustration of the method by which an automatic costing system would be devised. Greater potential is obtained, of course, if the systems philosophy is extended to include the Engineering and Manufacturing functions.

This new structuring concept will result in a better understanding, by the cost people, of product design logic and methods of manufacture. It will enable them to obtain:

- a) More effective cost analysis
- b) Better cost information for decision making purposes
- c) Simplified costing procedures

The financial area is one that probably offers the greatest opportunity and potential for economies and cost reduction. With a forward thinking systems group, TABSOL, and the powerful GE 225 computer these breakthroughs can be realities in the immediate future.



GENERAL COMPILER
SENTENCE FORM

PROGRAM		SAMPLE DECISION TABLE		DATE	
PROGRAMMER		COMPUTER		PAGE	
				OF	
SEQUENCE NUMBER					
5	PROCEDURE DIVISION.				
10	OPEN INPUT MASTER-FILE.				
15	GET~RECORD. READ MASTER~FILE RECORD IF END FILE GO TO END~RUN.				
20	IF FEMALE GO TO GET~RECORD.				
25	EXPERIENCE= 61 - YR~EMPLOYED + PREV~EXP.				
30	TABLE EXAMPLE. 3 CONDITIONS 2 ACTIONS 5 ROWS.				
35	LEVEL EQ EXPERIENCE TITLE I GO TO				
40	6 EQ 2 PROGRAMMER 1 TYPE~OUT				
45	7 EQ 3 PROGRAMMER OR ANALYST 2 "				
50	8 GR 3 ANALYST 3 "				
55	9 GR 4 ANALYST OR SR. ANALYST 4 "				
60	10 GR 4 SR. ANALYST 5 "				
65	GO TO GET~RECORD.				
70	TYPE~OUT. WRITE DEPARTMENT NAME TITLE LEVEL EXPERIENCE ON TYPEWRITER.				
75	TOTAL(I) = TOTAL(I) + 1.				
80	GO TO GET~RECORD.				
85	END~RUN. CLOSE MASTER~FILE.				
90	WRITE TOTAL(1) TOTAL(2) TOTAL(3) TOTAL(4) TOTAL(5) ON TYPEWRITER.				
95	STOP "END RUN".				

Computer Department, Phoenix, Arizona CA-13 (10-60)

V. TABSOL & GECOM

The culmination of the General Electric Company's progress in the use of TABSOL came in the union of TABSOL and GECOM. This development served to release the full power of the structure table.

Let us consider an example to develop an insight into the manner in which TABSOL is used in the General Compiler. The problem is to search a master employee file (recorded on magnetic tape) to determine the number of male employees who fall into the following job categories:

Job Level	Experience (Years)	Title
6	2	Programmer
7	3	Programmer or Analyst
8	More than 3	Analyst
9	More than 4	Analyst or Sr. Analyst
10	More than 4	Sr. Analyst

For each employee we find having these qualifications, we are to write his department number, name, title, level and experience on the computer's typewriter. At the end of the run the total for each category is also typed on the typewriter.

The core of this problem is the decision that must be made on the information stored in the records of the master file. These decisions are conveniently expressed above in narrative form. With only minor alteration, this form becomes the program statement of our problem. The table and sentences are punched into 80 column cards exactly as they appear in Figure 1. When this is done they may be given directly to the compiler for processing.

As illustrated in our example, General Compiler sentences may be used to support the logic of the table. These sentences accomplish the following:

OPEN - Sequence Number 10 - Declares that the MASTER-FILE is input and validates its tape labels.

READ - Sequence Number 15 - Delivers the next record from the MASTER-FILE and tests for an end-of-file sentinel. When this sentinel is detected, sequential program execution is interrupted, and control passes to the portion of the program labeled END-RUN.

IF - Sequence Number 20 - Eliminates those data records which contain information about female employees.

EXPERIENCE - Sequence Number 25 - Calculates the employee's total experience and assigns the value to the field named EXPERIENCE.

The word TABLE informs the compiler that it must process a decision table; EXAMPLE is a name or label which was given to the table. The size of the table is stated next by giving the number of conditions, actions and rows contained in the table. This information is used only by the compiler and is not executed by the compiled program.

Table execution begins at row 1 (sequence number 40). Using our narrative definition of a table, Row 1 is interpreted as follows: "IF the job LEVEL field equals (EQ) 6 AND the EXPERIENCE field equals (EQ) 2 years AND the employee's title is PROGRAMMER THEN assign the value 1 to the subscript I; GO TO the part of the program having the label TYPE-OUT."

If one of these conditions cannot be satisfied, row 2 is evaluated starting again with the left-most condition. Sequential execution of the rows continues until either all conditions in a given row are satisfied or all rows are exhausted. When the latter situation occurs, the sentence immediately following the table is executed. Proceeding from here, the sentences in our example accomplish the following:

GO - Sequence Number 65 - Interrupts sequential program execution and passes control to the part of the program labeled GET-RECORD.

WRITE - Sequence Number 70 - Writes the current contents of the DEPARTMENT, NAME, TITLE, LEVEL and EXPERIENCE fields on the computer's typewriter.

TOTAL (I) = TOTAL (I) + 1 - Sequence Number 75 - Increments the counter by one.

GO - Sequence Number 80 - Passes control to the part of the program labeled GET-RECORD.

CLOSE - Sequence Number 85 - Rewinds the MASTER - FILE and performs the file's closing conventions.

WRITE - Sequence Number 90 - Writes totals for each category on the typewriter.

STOP - Sequence Number 95 - Terminates processing and writes the word END-RUN on the typewriter.

By General Compiler standards this example represents relatively simple conditions and actions. In formulating these entries, the programmer may take full advantage of the compiler's capabilities.

Figure 2 and Figure 3 show how the manufacturing planning tables developed in Section IV-A would appear in the GECOM format.

For more detailed explanations of the conventions and manner in which conditions and actions may be formed and entered in tables as well as a detailed

explanation of the General Compiler, refer to the following Computer Dept. publications.

- a) TABSOL 225 - Reference Manual
- b) General Compiler Manual - 225

Keep in mind the relative ease with which the table was entered for the computer operation. There was no translation process from the System Analyst's language to the computer language. The fantastic power is that functional specialists can now write tables directly for computer entry'

PROGRAM		ROTOR STACK & PRESS OPERATION		DATE																																																																											
PROGRAMMER		COMPUTER		PAGE 1 of 2																																																																											
1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31	32	33	34	35	36	37	38	39	40	41	42	43	44	45	46	47	48	49	50	51	52	53	54	55	56	57	58	59	60	61	62	63	64	65	66	67	68	69	70	71	72	73	74	75	76	77	78	79	80
SEQUENCE NUMBER																																																																															
5		PROCEDURE DIVISION.																																																																													
10		OPEN INPUT MASTER~FILE.																																																																													
15		GET~RECORD. READ MASTER~FILE RECORD IF END FILE GO TO END~RUN.																																																																													
20		TABLE ONE. 1 CONDITION 2ACTIONS 2 ROWS.																																																																													
25		POLES	LAMID	GO TO																																																																											
30		EQ 2	5	TABLE TWO																																																																											
35		GR 2		TABLE TEN																																																																											
40		TABLE TWO. 2 CONDITIONS 5 ACTIONS 5 ROWS.																																																																													
45		TIME~1. OP~TIME = 100 + 5 * STK~HT.																																																																													
50		TIME~2. OP~TIME = 110 + 5 * STK~HT.																																																																													
55		BEGIN.																																																																													
60		STK~HT	STK~HT	ARBOR	COLLAR~THICK	CODE	PERFORM	SETUP																																																																							
65		NLS 2	LS 5	7	7-STK~HT	1	TIME~1	20																																																																							
70		NLS 5	LS 10	12	12-STK~HT	1	TIME~1	20																																																																							
75		NLS 10	LS 15	17	17-STK~HT	1	TIME~2	25																																																																							
80		NLS 15	LS 20	22	22-STK~HT	1	TIME~2	25																																																																							
85		NLS 20	LS 25	27	27-STK~HT	1	TIME~2	25																																																																							

Computer Department, Phoenix, Arizona CA-13 (10-60)



GENERAL COMPILER
SENTENCE FORM

PROGRAM ROTOR STACK & PRESS OPERATION										DATE																																																																															
PROGRAMMER										COMPUTER																																																																															
										PAGE 2 of 2																																																																															
1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31	32	33	34	35	36	37	38	39	40	41	42	43	44	45	46	47	48	49	50	51	52	53	54	55	56	57	58	59	60	61	62	63	64	65	66	67	68	69	70	71	72	73	74	75	76	77	78	79	80										
SEQUENCE NUMBER																																																																																									
90	TABLE THREE. 2 CONDITIONS 1 ACTION 3 ROWS.																																																																																								
95	LAMOD	LAMOD	COLLAR~OD																																																																																						
100	GR 5	NGR 10	7.00																																																																																						
105	GR 10	NGR 15	10.00																																																																																						
110	GR 15		13.00																																																																																						
115	TABLE FOUR. 1 CONDITION 2 ACTIONS 4 ROWS.																																																																																								
120	CODE	OP~RATE	SETUP~RATE																																																																																						
125	1	.020	.015																																																																																						
130	2	.025	.015																																																																																						
135	3	.030	.015																																																																																						
140	4	.035	.015																																																																																						
145	PRICE~CALC. PRICE = OP~TIME*OP~RATE + SETUP*SETUP~RATE																																																																																								
150	GO TO OUTPUT~1.																																																																																								

Computer Department, Phoenix, Arizona CA-13 (10-60)

VI. CONCLUSION

Examples of the application of TABSOL applied to manufacturing, engineering and financial problems were presented to illustrate the power and scope of the structure table concept.

The fields discussed are by no means the extent of potential application - TABSOL may be applied to all functional operations. In manufacturing, in addition to Quality Control and Operation Planning previously discussed, we can structure:

- a) the decision logic for inventory control including the ABC inventory techniques.
- b) production scheduling and dispatching with many variables and decision requirements.
- c) shipping and traffic control to determine preferred shipping instructions depending on cost, time and urgency.

In Marketing, structure tables have been used to formulate propositions. Such tables show the relation between market requirements and resulting engineering decisions, together with estimated cost. In processing these tables, the cost is accumulated as the engineering proceeds from table to table, resulting in a final set of specifications for the proposition and the estimated total cost.

A proven result of the application of structure tables to the business decision process is the advantage of providing a clear understanding of the interaction of the complex forces in every-day business life.

There are many other benefits derived from learning, analyzing, formulating, and recording the decision logic for a structure table application including the following:

- a) Structure tables force a logical step by step analysis of the decision.
- b) Structure tables force consideration of all logical alternatives.
- c) Structure tables are easily understood and thus form an excellent basis for communication between functional specialists and systems analysts.
- d) Structure tables can be written by the functional specialist for direct input to the GE 225 computer without having to go through the computer programmer, thus reducing computer application costs.
- e) Structure tables are easy to maintain, and a system may be easily revised by changing a single value in a single table. In some manual systems inaccuracy is tolerated due to the expense of updated files. This inaccuracy is no longer necessary.
- f) The GE 225 electronic computer offers unsurpassed accuracy, ability and economy in the processing of structure table logic.

The greatest potential of this new language concept lies in our ability to apply it toward the development of a completely integrated business system. The enormous number of daily, routine decisions made by trained and talented personnel are now within the range of mechanization.

The task of completely structuring this logic is the challenge facing our computer users today. When this challenge is met we can truly expect to see a completely automatic business system. An automatic system to take an order received as input and transfer it to a finished product as output, without manual intervention will be the normal business practice. TABSOL is the key to this tremendous progress and the GE 225 Electronic Data Processing System is today's answer to the challenges it offers.