

DPHQ Systems Engineering Services
White Plains, N. Y.
August 21, 1961

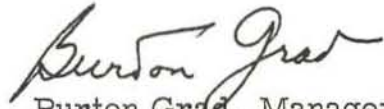
Subject: Tabular Techniques Distribution #4, and
Confidential Nature of Releases

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.

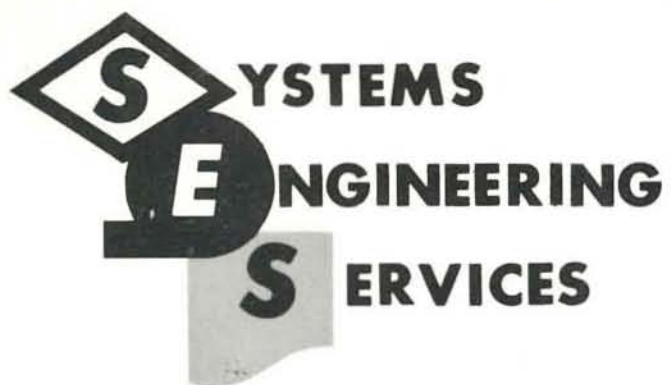
Many consider tables to be useful in performing the analysis necessary in designing a new system; some maintain that they are more powerful as a programming tool; others claim that the greatest benefit is derived when used as a standard documentation technique. But regardless of the area of use, we feel 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,
Data Processing Division Headquarters,
White Plains, N.Y.

Reprinted from DATAMATION Magazine, July 1961

An F. D. Thompson Publication

Tabular form has shown promise of being an effective way to organize and present decision logic for systems analysis and computer programming. Experience to date clearly indicates the need for further exploration and development of tabular form to determine its range of application and assess its future potential. This report has the dual purpose of sketching the historical background on the development of tabular form, and indicating its possible advantages.

TABULAR FORM IN DECISION LOGIC

by BURTON GRAD, IBM Corporation,
Data Processing Division Headquarters,
White Plains, N.Y.

Glancing around the office, I can see three young women busily engaged in the various duties of a typical work day. Let me tell you about them. Blond Marilyn is a chatterbox. Penelope and Theresa enjoy going to the movies. Marilyn is married, but the other two are single. Penelope has an attractive figure, while Marilyn is somewhat on the plump side. Theresa's quiet moods contrast to Penelope's happy ones, but they both seem to enjoy life in native Manhattan. Marilyn has dimples; Theresa may be recognized by her amber eyes and red hair. Unlike the others, Marilyn prefers Shakespeare and country living in Chappaqua.

Without looking back, can you recall all of Penelope's characteristics? Do you have a clear image of each girl and know what data is missing or where there are inconsistencies? To help answer these questions, let's rearrange the information. Displayed in tabular form, it would appear as in Figure 1:

Name	Marilyn	Penelope	Theresa
Marital Status	Married	Single	Single
Hair Color	Blond		Red
Figure	Plump	Attractive	
Enjoys Movies		Yes	Yes
Prefers Shakespeare	Yes	No	No
Residence	Chappaqua	Manhattan	Manhattan
Features	Dimples		Amber Eyes
Characteristics	Chatterbox	Happy	Quiet

Figure 1

From this illustration, some of the advantages of tables over narrative style for comparative data display can be readily appreciated: **Conciseness and clarity** is achieved by classifying data; **Completeness** is insured by revealing areas where information is missing; **Meaningful relationships** are recognized quickly and easily with the two dimensional structure.

While recognizing these advantages many will point out that tables are merely a systematic way to present static data. Do they have a worthwhile function in a more dynamic situation—that of decision making? Would tables be valuable in systems analysis and computer programming? Before we explore some preliminary answers to these questions, let's look at a brief history of tables.

universality of tables

Tables, whether statistical, financial, or analytical, have gained widespread recognition; they seem to be a natural form for expressing relationships among variable factors where there are many possible patterns for arranging the significant information. This fact is substantiated by the profusion of examples in everyday life:

The ubiquitous government reports with ponderous breakdowns of the GNP or a simple recap on whooping crane birth rates and population.

The multiplicity of financial reports showing the status and growth of businesses.

The economic forecasts of things to come ranging from hula-hoop production to manned satellites in the burgeoning 60's and beyond.

The daily scratch sheet, the box scores of runs, hits and errors for the latest baseball games, and the highs, lows, and closing prices for stocks — all in the local newspaper.

And the list grows.

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>< 35</u>	<u>≥ 25</u> <u>< 35</u>	...	<u>≥ 65</u>
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

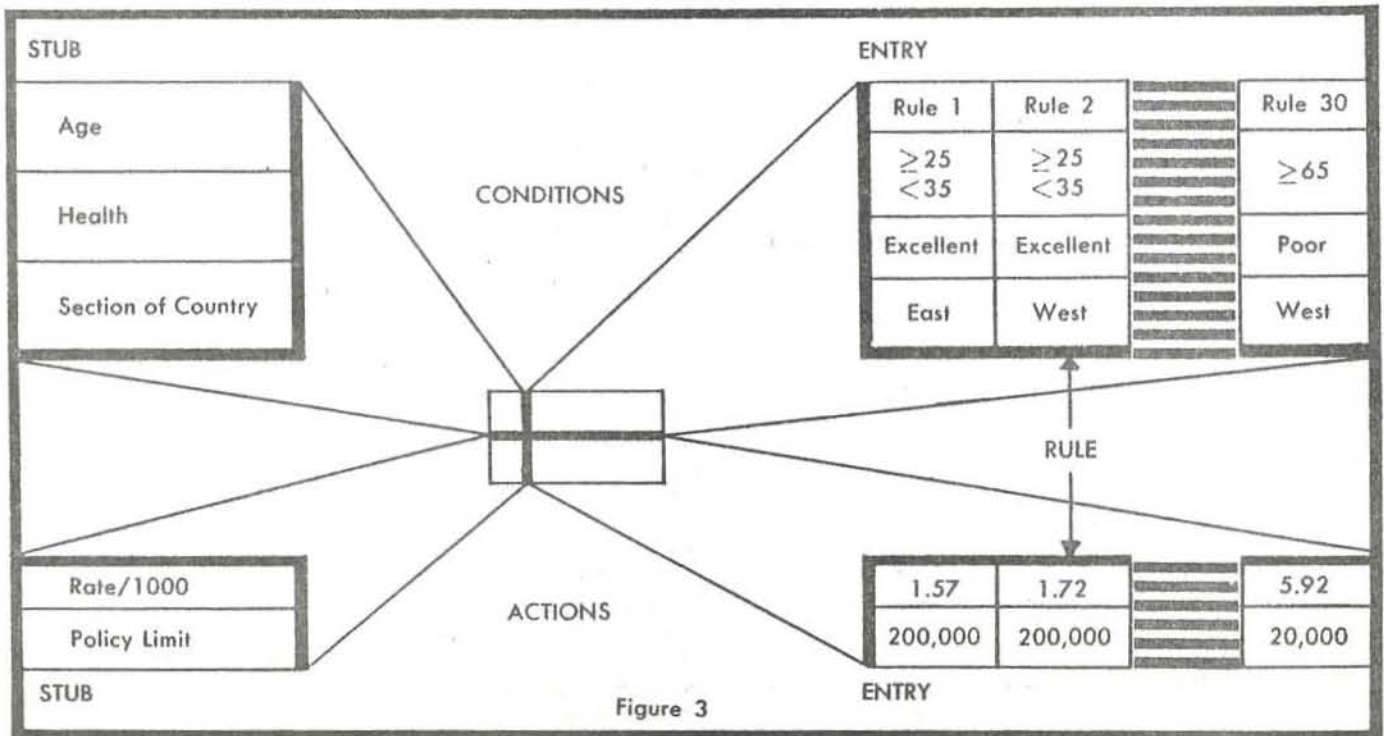


Figure 3

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 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:

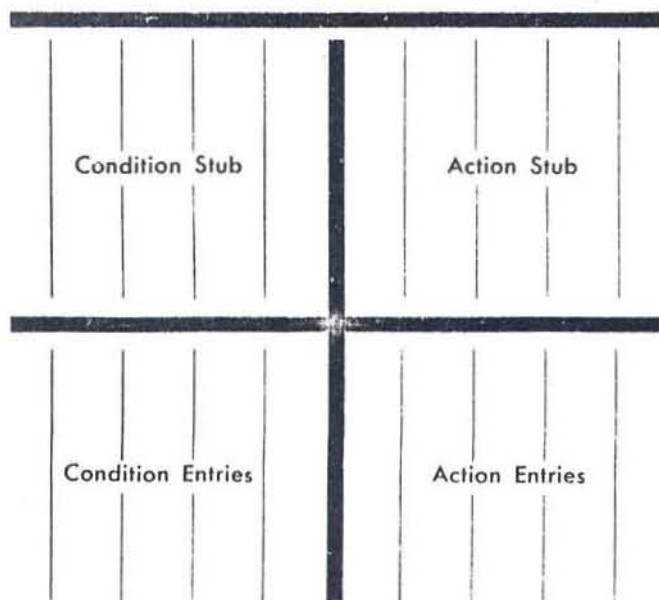


Figure 5

Y = yes

N = no

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

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

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

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.

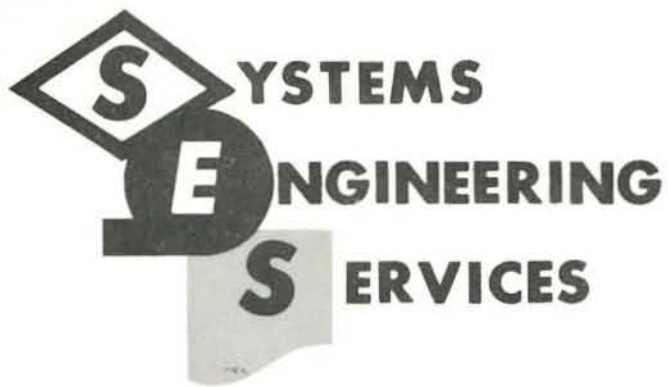
What then of the future? Would it be desirable to directly incorporate tabular form into existing language processors such as Autocoder, FORTRAN, Commercial Translator, or COBOL, to describe complex decision procedures with decision tables? Would this approach significantly improve logical analysis?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



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 - 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.

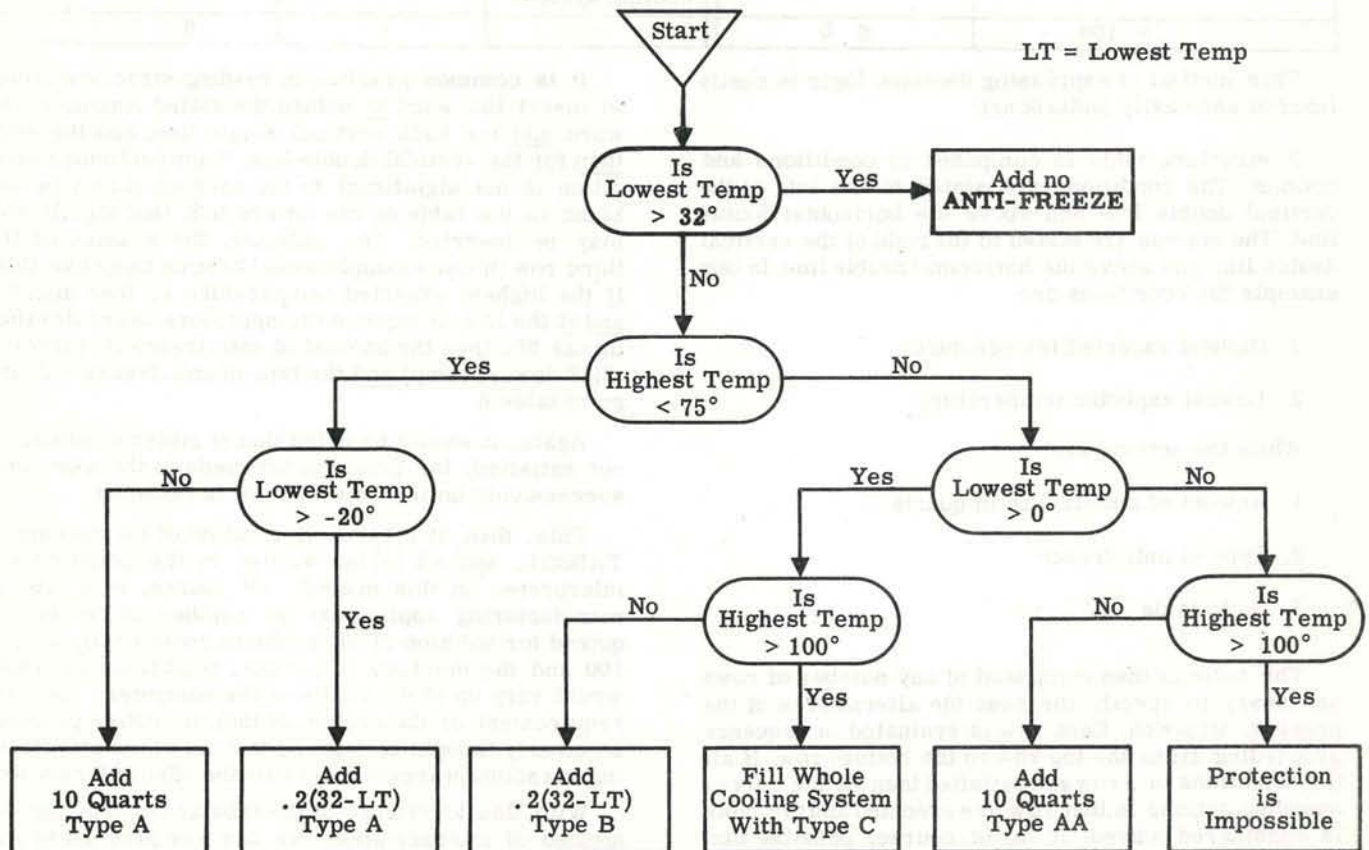
III. HOW TO READ STRUCTURE TABLES

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				
	Stack ⑥ Stacks ⑦ inches				
2					

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:

- Inspection points
- Dimensional characteristic of each inspection point
- Required inspection tools
- Tolerances permitted
- 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 \geq	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 \geq	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 \geq	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 \geq	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.

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.

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

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.

PROGRAM		SAMPLE DECISION TABLE																																													DATE																																																																				
PROGRAMMER		COMPUTER																																													PAGE																																																																				
																																															OF																																																																				
		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	IF FEMALE GO TO GET~RECORD.																																																																																																																	
	25	EXPERIENCE = 61 - YR~EMPLOYED + PREV~EXP.																																																																																																																	
	30	TABLE EXAMPLE. 3 CONDITIONS 2 ACTIONS 5 ROWS.																																																																																																																	
	35	<table border="1" style="width: 100%; border-collapse: collapse;"> <thead> <tr> <th style="width: 10%;">LEVEL</th> <th style="width: 10%;">EQ</th> <th style="width: 20%;">EXPERIENCE</th> <th style="width: 20%;">TITLE</th> <th style="width: 5%;">I</th> <th style="width: 35%;">GO TO</th> </tr> </thead> <tbody> <tr> <td>6</td> <td></td> <td>EQ 2</td> <td>PROGRAMMER</td> <td>1</td> <td>TYPE~OUT</td> </tr> <tr> <td>7</td> <td></td> <td>EQ 3</td> <td>PROGRAMMER OR ANALYST</td> <td>2</td> <td>"</td> </tr> <tr> <td>8</td> <td></td> <td>GR 3</td> <td>ANALYST</td> <td>3</td> <td>"</td> </tr> <tr> <td>9</td> <td></td> <td>GR 4</td> <td>ANALYST OR SR. ANALYST</td> <td>4</td> <td>"</td> </tr> <tr> <td>10</td> <td></td> <td>GR 4</td> <td>SR. ANALYST</td> <td>5</td> <td>"</td> </tr> </tbody> </table>																																																																														LEVEL	EQ	EXPERIENCE	TITLE	I	GO TO	6		EQ 2	PROGRAMMER	1	TYPE~OUT	7		EQ 3	PROGRAMMER OR ANALYST	2	"	8		GR 3	ANALYST	3	"	9		GR 4	ANALYST OR SR. ANALYST	4	"	10		GR 4	SR. ANALYST	5	"
LEVEL	EQ	EXPERIENCE	TITLE	I	GO TO																																																																																																														
6		EQ 2	PROGRAMMER	1	TYPE~OUT																																																																																																														
7		EQ 3	PROGRAMMER OR ANALYST	2	"																																																																																																														
8		GR 3	ANALYST	3	"																																																																																																														
9		GR 4	ANALYST OR SR. ANALYST	4	"																																																																																																														
10		GR 4	SR. ANALYST	5	"																																																																																																														
	65	GO TO GET~RECORD.																																																																																																																	
	70	TYPE~OUT. WRITE DEPARTMENT NAME TITLE LEVEL EXPERIENCE ON TYPEWRITER.																																																																																																																	
	75	TOTAL(1) = 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".																																																																																																																	

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 2 ACTIONS 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																			



GENERAL COMPILER
SENTENCE FORM

PROGRAM		DATE	
PROGRAMMER		COMPUTER	PAGE
ROTOR STACK & PRESS OPERATION			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	L A M O D L A M O D C O L L A R ~ O D		
100	G R 5 N G R 10 7.00		
105	G R 10 N G R 15 10.00		
110	G R 15 13.00		
115	TABLE FOUR. 1 CONDITION 2 ACTIONS 4 ROWS.		
120	C O D E O P ~ R A T E S E T U P ~ R A T E		
125	1 .020 .015		
130	2 .025 .015		
135	3 .030 .015		
140	4 .035 .015		
145	P R I C E ~ C A L C . P R I C E = O P ~ T I M E * O P ~ R A T E + S E T U P * S E T U P ~ R A T E		
150	G O T O O U T P U T ~ 1 .		