SYSTEMS ENGINEERING SERVICES

CLEARINGHOUSE REPORT

DECISION TABLES

A PRELIMINARY REFERENCE MANUAL

September 1961
Ref. No. 1 J 1

Orren Y. Evans
DP - Western Region

INTERNATIONAL BUSINESS MACHINES CORPORATION

White Plains, New York

PREFACE

This manual, DECISION TABLES, is a sequel to the paper, "An Advanced Analysis Method for Integrated Electronic Data Processing", written in the fall of 1959. The paper was first published by the National Machine Accountants Association of Long Beach in March, 1960. Later in 1960, it was published by IBM in a condensed version as a General Information Manual (Form Number F20-8047).

Since these publications and reviews in the Data Processing Digest and Datamation, there have been numerous requests for more information on the method. The frequency of these requests has prompted the writing of this manual.

The manual is not written with the intention of making a processor available for mechanizing its implementation. It is not intended to obligate IBM to provide follow-on processors.

The purpose of this manual is to stimulate further interest in the DECISION TABLES to such a degree as to prompt more persons to experiment with them and prove or disprove their utility.

ACKNOWLEDGMENTS

There have been so many people who have contributed to ideas expressed in this document as to preclude naming all of them. The two who have made major contributions are Mr. Burton Grad, Manager of IBM Systems Engineering Services, and Mr. Abner Copeland, Manager of Hunt Foods Planning and Analysis.

I have drawn heavily from IBM COMMERCIAL TRANSLATOR and COBOL, '61, since the language in these manuals is and will be familiar to many people. I have modified their language in (1) adding dimension and (2) when necessary, as a convenience to the Decision Table format.

In addition, I have used ideas from the "Report Program Generator for the IBM 1401" for specifications of the report generation features.

In accordance with the requirements of the official government manual describing COBOL-1961, the following extract from that manual is presented for the information and guidance of the user:

"This publication is based on the COBOL System developed in 1959 by a committee composed of government users and computer manufacturers. The organizations participating in the original development were:

Air Materiel Command, United States Air Force
Bureau of Standards, United States Department of Commerce
Burroughs Corporation
David Taylor Model Basin, Bureau of Ships, United States Navy
Electronic Data Processing Division, Minneapolis-Honeywell Regulator Company
International Business Machines Corporation
Radio Corporation of America
Sylvania Electric Products Inc.
UNIVAC Division of Sperry Rand Corporation

"In addition to the organizations listed above, the following other organizations participated in the work of the Maintenance Group:

Allstate Insurance Company
The Bendix Corporation Computer Division
Control Data Corporation
E. I. du Pont de Nemours and Company
General Electric Company
General Motors Corporation

ACKNOWLEDGEMENTS

(Continued)

Lockheed Aircraft Corporation
The National Cash Register Company
Philco Corporation
Standard Oil Company (New Jersey)
United States Steel Corporation

"This COBOL-61 manual is the result of contributions made by all of the above-mentioned organizations. No warranty, expressed or implied, is made by any contributor or by the committee as to the accuracy and functioning of the programming system and language. Moreover, no responsibility is assumed by any contributor, or by the committee, in connection therewith.

"It is reasonable to expect that many improvements and additions will be made to COBOL. Every effort will be made to insure that improvements and corrections will be made in an orderly fashion with due recognition of existing users' investments in programming. However, this protection can be positively assured only by individual implementors.

"Procedures have been established for the maintenance of COBOL. Inquiries concerning the procedures and the methods for proposing changes should be directed to the Executive Committee of the Conference on Data Systems Languages.

"The authors and copyright holders of the copyrighted material used herein: FLOW-MATIC (Trade-mark of Sperry Rand Corporation) Programming for the UNIVAC I and II Data Automation Systems © 1958, 1959, Sperry Rand Corporation; IBM Commercial Translator, Form No. F28-8013 copyrighted 1959 by IBM, have specifically authorized the use of this material, in whole or in part, in the COBOL 61 specifications. Such authorization extends to the reproduction and use of COBOL specifications in programming manuals or similar publications.

"Any organization interested in reproducing the COBOL report and initial specifications in whole or in part using ideas taken from this report or utilizing this report as the basis for an instruction manual or any other purpose is free to do so. However, all such organizations are requested to reproduce this section as part of the introduction to the document. Those using a short passage as in a book review are requested to mention 'COBOL' in acknowledgement of the source but need not quote the entire section."

CONTENTS

Chapter 1: GENERAL DESCRIPTION OF THE DECISION TABLE SYSTEM

INTRODUCTION A definite need exists for a method of determining and documenting systems requirements. Systems requirements should be differentiated from procedures to accomplish the systems requirements. We have tended to use procedures for stating and accomplishing systems requirements and thus have hampered our understanding of the true systems requirements and our consideration of alternate system solutions.

Documentation of systems requirements can be made at many levels. There are three major levels apparent (with multiple levels possible within each).

1. Machine run (or clerical activity) requirements and solution definition. This level of definition describes in detail the solution including all machine (Process media) considerations.

2. Machine run (or clerical activity) requirements with minimum emphasis on the solution. After a systems flow has been determined as to machine runs and decisions made as to what generally is included in each run the analyst defines the decision logic of the run relating output to input. He does this with little, if any, regard to machine considerations of accomplishing the decision logic.

3. Systems definition at the systems level which is virtually independent of the systems solution (does not consider individual machine runs or methods of accomplishing the decision logic). Definition at this level includes not only decision logic relating input to output, but systems constraints and acceptance criteria; i.e. systems operational data resources for designing, implementing and operating systems, response times in the system, methods of evaluating alternate system solutions etc.

Summarizing, we are faced with problem definition and problem solution (the what and the how). This manual offers a method of documenting at any of these levels, but it is a compromise between the what and the how. Procedure step sequencing is almost non-existent within a decision table. However, table sequence control is very specific and thus is, to a degree, procedure oriented.

Decision Table format, syntax and language have been kept simple enough to learn and use easily but is sufficiently sophisticated to permit high level data manipulation commands. The decision table documentation does not include provisions for systems resources, constraints and scoreboard (method of evaluating different system solutions).

PLAN OF MANUAL It is assumed that the reader has knowledge of the essential characteristics of Data Processing Systems. The manual is written as a base language (point of departure) for using decision tables. The language is not all inclusive and in many instances has been arbitrary in the interest of simplicity. It provides a language that interested persons can use to experiment with decision tables in documenting problem definitions. Let us attempt some broad field experience in using decision tables to describe problems at several levels and then refine the language.

This language is very rigorous in order that it may be used at the detail level of documentation. People experimenting at higher levels of man to man communication can adjust this rigor to their needs.

Chapter 2: THE STRUCTURE OF DECISION TABLES

INTRODUCTION This chapter is concerned with a brief introduction to the general concepts structure and elements of decision tables. Detail explanation of procedure description using decision tables is found in Chapter 4.

DECISION TABLE

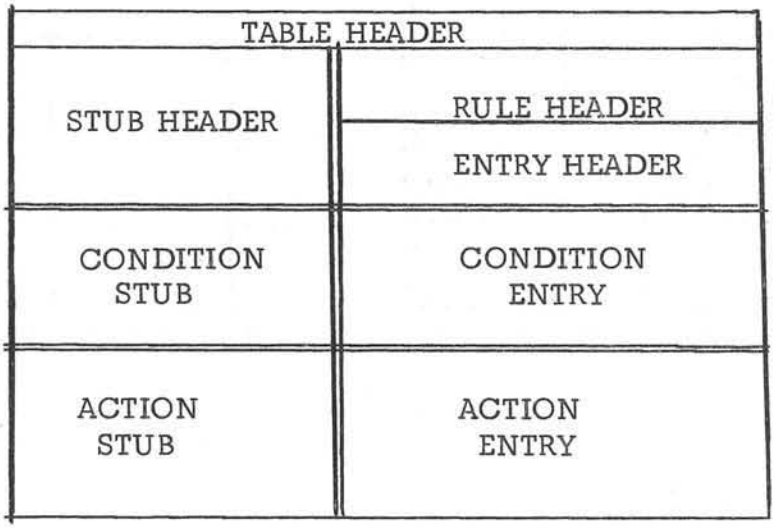


FIGURE 2.1

TABLE ELEMENTS The main elements of the decision table are noted in Figure 2.1 above. The TABLE HEADER contains identification data such as title, prepared by, date and system. The RULE HEADER is, primarily, for identifying each rule with a number so that it can be referenced in written and oral communication. The STUB HEADER identifies columns (verb, operand, operator) for condition and action data. The ENTRY HEADER identifies columns (operator, operand, action sequence) for further data of conditions and actions pertinent to a particular rule. The vertical double line separates the STUB and ENTRY data. The double horizontal line separates the CONDITIONS and ACTIONS. These elements can be best clarified by discussing an example (FIGURE 2.2)

TABLE: CREDIT CHECK								
LINE NR.	VERB	OPERAND	OP	OPERAND	RULE 1 OPERAND	RULE 2 OPERAND	RULE 3 OPERAND	RULE 4 OPERAND
1		SPECIAL CLEAR	EQ		1	O	O	O
2		CREDIT LIMIT	GE	OK*		Y	N	N
3		PAY EXPER-IENCE	EQ				GOOD	BAD
4	MOVE	'APPRO-VED'	TO	ORDER STATUS	X	X	X	
5	MOVE	'REJECT'	TO	ORDER STATUS				X

*OK = In process amount + Accounts Receivable amount + Order amount

FIGURE 2.2

INTERPRETATION OF RULES Rule 1 reads: If special clearance is EQ (equal) to 1, then move the word 'approved' to order status. In less precise terminology rule one means: If special clearance is obtained, approve the order.

Rule 2 reads: If special clearance is EQ (equal) to O and the credit limit is GE (greater than or equal to) OK then move the word 'approved' to order status. Less precisely, rule two means: If special clearance is not obtained and the credit limit is OK then approve the order.

Rule 3 reads: If special clearance is EQ (equal) to O and the credit limit is not GE (not greater than or not equal to) OK and pay experience is EQ (equal to) GOOD, then move the word 'approved' to order status. Less precisely, rule three means: If special clearance is not obtained and the credit limit is not OK and the pay experience is good, then approve the order.

Rule 4 reads: If special clearance is EQ (equal) to O and the credit limit is not GE (not greater than or not equal to) OK, and pay experience is BAD, then move the word "reject" to order status. Less precisely, rule four means: If special clearance is not obtained and credit limit is not OK and pay experience is bad, then reject the order.

A RULE consists of a unique group of condition(s) and the action(s) to be taken when those condition(s) exist. In FIGURE 2.2 each of the columns headed by RULE 1, RULE 2, RULE 3 and RULE 4 depict vertical combinations of conditions and actions.

A CONDITION is a state of existence of a specific piece of information. The state of existence may be its presence or absence a specific value or range of values; or its relationship to other data; or combinations of these. Structurally, in decision tables a condition consists of two factors (1) the data in the STUB and (2) the data in a rule ENTRY. For instance, the condition in line 3 of rule 3 "pay experience equals good" is the complete condition. The condition in line 3 of rule 4 is "pay experience equals bad." A line stub data serves as a common factor for multiple conditions on that line.

ACTIONS are commands which perform operations (movement, arithmetic, etc.) on and with data as well as control the sequence of considering tables. The structure of actions in decision tables is similar to the structure of conditions explained above.

A LIMITED ENTRY CONDITION is shown in line 2. The entire condition (credit limit is greater than or equal to OK) is stated in the STUB. Data in the ENTRY of each rule, for line 2, is limited to "Y", "N" or left blank.

Y means yes the condition must be satisfied.

N means no the condition must not be satisfied.

Left blank means the condition is irrelevant.

A LIMITED ENTRY ACTION is shown in line 4. The entire action (move the word "approved" to order status) is stated in the STUB. Data in the ENTRY of each rule, for line 4, is limited to "X" or left blank.

X means take this action.

Left blank means do not take this action.

EXTENDED ENTRIES have a portion of the condition or action stated in the ENTRY and the remainder in the STUB. See line 3.

MIXED ENTRIES A rule or a table may contain both limited and extended entries whichever is more convenient for a particular condition or action.

ARITHMETIC EXPRESSION In line 2 the condition is "credit limit is GE (greater than or equal to) OK." The word "OK" is the name of an arithmetic expression which is defined just below the table as:

OK = in process amount + accounts receivable amount + order amount.

Since long expressions do not fit well in the decision tables, they are named and the name is placed in the table while the expression is defined elsewhere. Each time an arithmetic expression is called for in a condition or action, its value is recomputed. The value of an expression does not persist.

OPERANDS in Figure 2.2 contain:

- name of an arithmetic expression
- limited entries
- names of variable data fields
- names of conditions
- literals

Limited entries and arithmetic expressions have been explained. Names of variable data fields are illustrated in FIGURE 2.2 by

- special clearance
- credit limit
- pay experience
- order status

Names must contain at least one alphabetic character. The name of a field is quite different from the value of the field. To illustrate the difference:

FIELD NAME	FIELD VALUE
credit limit	$10,000
order amount	$ 1,500
customer	ABC Manufacturing Company

The name of a condition is illustrated in line 3, rules 3 and 4. This concept will be explained in more detail in Chapters 3 and 5. Briefly, the field name "pay experience" can have a code of 1 or 0 where

1 means favorable
0 means unfavorable

In the data description, provision is made to name the possible code values of a field. The names of the codes are called CONDITION NAMES. In FIGURE 2.2 codes are named:

FIELD NAME	CODE VALUE	CONDITION NAME
pay experience	1	good
	0	bad

Literals are illustrated in the entries (1,0,0,0) of line one and the first stub operands ("approved" , "reject") of lines 4 and 5. A literal is an unnamed constant. It is the value rather than a name. Alphameric literals are identified by quote marks. Alphameric literals may contain any character except quote marks. Numeric literals contain only numerals and do not require quote marks for identification.

This chapter has presented the concept and structure of Decision Tables and an indication of its syntax and vocabulary.

Chapter 3: THE STRUCTURE OF THE LANGUAGE

The structure of the Decision Table language has a basic vocabulaty consisting of words and symbols and implied words in conditional action statements. (These are the interpretation of rules as discussed in Chapter 2.) It has a set of rules by which words and symbols may be combined to express meaning. In addition, it has punctuation rules to inject clarity into groups of words and symbols.

The language consists primarily of nouns and verbs. The verbs are explicitly defined and named, and are a part of the basic vocabulary. A few nouns are thus defined and named but, for the most part, the nouns are data names and are generated and defined and placed into the decision logic by the user.

CHARACTER SET. Words and symbols are the basic units of the language but are, in turn, composed of individual letters, numerals and special characters (basic character set). This set consists of the 26 letters of the alphabet, the ten numerals from 0 through 9, and the special characters shown in the table below.

Name	Character
Blank	
Plus sign	+
Minus sign	-
Multiplication sign	x
Division sign	/
Left parenthesis	(
Right parenthesis	)
Comma	,
Period and decimal point	.
Dollar sign	$
Equal sign	=
Quotation mark	'

NAMES. Most of the words will be nouns. A noun, in the strictest sense, is a NAME, and the user will find it useful to accept that definition and all its implications. He will prepare decision tables for handling data, but will refer to the tables and data by name.

KINDS OF NAMES. Most of the names will fall into one of four kinds: data names, condition names, table names and expression names. These names are entered in decision table columns headed OPERAND.

DATA NAMES are names given to the data used in defining a problem or system. Data names are assigned to KINDS of data; not (except in the case of constants) to specific values. Thus, a name such as INTEREST.RATE would not refer to a specific interest rate, but to a class of data known as interest rates.

Data-names may be assigned to single classes of data or to groups of data items. For example, the name PAYROLL.RECORD would probably refer to a group of individual items having such names as EMPLOYEE.NAME, EMPLOYEE.NUMBER, HOURLY.RATE, and so on. It is important to recognize that, in general, the name refers not to any specific value, but only to the kind of data.

Data-names are invented and assigned to data at the discretion of the user, following the rules given below which govern the formation of names. All data referred to in the problem definition must be named, but this does not mean that all subdivisions of data must be named. For example, if the user wishes to refer to a date, he will have to give it a name, but he does not have to name the component parts, such as the day, the month, or the year, unless he wishes to refer to them individually.

The general category of data-names may be broken down, for convenience, into record-names, item-names, expression-names, named constants, and so on. The meaning of these names will be explained later in this manual.

CONDITION NAMES are names assigned to specific values of a data name. This naming is for the convenience of the user. The naming is at the discretion of the user, but is usually done such that the name has a mnemonic meaning associated with a specific value of the data name. In Chapter 2 an illustration was described on page 2.5. The data name was PAY.EXPERIENCE which has two possible values, 1 or 0. Where 1 means favorable and 0 means unfavorable. The following table shows how the condition naming can give a clue as to its value meaning and thus aid in the communication value of the decision table rules.

Value	Meaning	Condition Name
0	Pay.Experience is unfavorable	Bad
1	Pay.Experience is favorable	Good

Rules 3 and 4 of Decision Table, Credit Check, FIGURE 2.2, illustrate its use in decision logic.

TABLE NAMES are names assigned to individual tables so they can be referred to by table sequence control verbs and other portions of the problem definition.

EXPRESSION NAMES are names of arithmetic expressions. Since long arithmetic expressions do not fit in the decision table, they are named for convenience and the expression is defined in the data description.

FORMATION OF NAMES. Names are formed by combining any of the characters from the basic list of alphabetic characters, numerals and the period, subject to the following rules:

1. Names must not contain blanks.

2. Names, other than table names, must contain at least one alphabetic character. Table names may consist entirely of numerals if the user so desires.

3. They should be kept reasonably short since the operand space in the table is quite small. Names should be limited to 6 or less characters.

4. They may neither begin nor end with a period. However, "imbedded" periods may be used within the name for the sake of readability.

5. They may be "qualified" (to make them unique within the problem definition) by the use of other names. This is explained below under the heading "compound names."

COMPOUND NAMES. In many cases a problem definition will contain duplicate names. This often happens when an input file is "updated" to produce an output file, since each file will usually contain the same kinds of records.

Suppose that an input record is named IN.MAS and an output record is called OT.MAS. Suppose, further, that each record contains two dates, one called ORD.DAT, the other called SHP.DAT.

If the problem definition involves both kinds of records, it would not be possible to distinguish readily between the two ORD.DAT names and the two SHP.DAT names. All four names would be defined in the data description (see Chapter 5), which gives the system the information it needs to locate individual items of data. To indicate which of the ORD.DAT (or SHP.DAT) names is meant, however, each such name can be "qualified," or "compounded," when used in a decision table. That is, the name of a larger data item of which it is a part can be added to the name to identify it. Thus, IN.MAS ORD.DAT would be clearly distinguishable from OUT.MAS ORD.DAT. Names qualified in this manner are referred to as COMPOUND NAMES.

When names are to be compounded, the following rules apply:

1. Each name must be separated from the next by at least one blank space. (This distinguishes between compound and simple names, since simple names may not contain blanks.)

2. The names must be written in increasing order from the general to the specific. (If the reader is familiar with the concept of level numbers, as discussed in Chapter 5 of this manual, he will note that this means that the names must be listed in order from the lowest to the highest level number.)

3. No qualifying names are required that do not contribute to the uniqueness of the compound name. Thus, in the example given, if there were only one date names in each of the input and output files, e.g., an ORD.DAT, but no SHP.DAT--it would not be necessary to use the name ORD.DAT in forming the compound name; the names IN.MAS DAT and OT.MAS DAT would suffice.

The organization and structure of data for use in a data processing system is further discussed in Chapter 5, entitled "Data Description." The reader is referred, in particular, to the discussion of level numbers beginning on page 5.5.

PLACING NAMES IN THE PROBLEM DEFINITION. The reader has seen that this system uses names as a convenient--in fact, indispensable--means of identifying data, decision tables, and conditions. It is now necessary to indicate how each name is placed in the problem definition in a way that permits the system to connect it with the item to which it refers.

A problem definition consists primarily of decision tables and "data description." The first of these is made up of rules, which is the actual decision logic. The second consists of data description statements which show the organization and nature of the data so that it can be located and used when needed. These two sections are discussed in Chapters 4 and 5.

All statements must be written in a specified format. Two columnar forms have been prepared for this purpose. One is used for decision tables and the other for data description statements. The first is described in Chapter 4, the second in Chapter 5.

For data-names and condition-names, as will be seen in Chapter 5, the system must know whether the data is numeric or whether it contains alphabetic characters. It must know where decimal points, if any, are to be placed, where to print dollar signs, and so on. There are a number of such details which must be specified. It would have been possible to set up rules for describing the data in the decision tables, but this would have been inefficient, since the description of each item would have had to be repeated each time its name appeared. Since the description is placed in a separate section, however, each name need be described only once, regardless of the number of times it is used in the problem definition.

It follows that each data-name and each condition-name used in the procedure description must be properly accounted for in the data description, following the rules given in Chapter 5. Once this has been done, the programmer is free to refer to the name repeatedly throughout the decision tables.

CONSTANTS. It has been emphasized that the data-names used in the system generally refer to kinds of data, not to specific values. The actual values represented by most data-names are assumed to be variable, and they will either be entered into the systems as parts of input files or will be computed at some point when the system is in operation.

However, the user will often find it useful to be able to place a specific fixed value into the program definition instead of having to read it in as data. For example, if a firm allows a discount on its bills, the discount will usually be figured as a fixed percentage. The routine for computing the discount, therefore, does not require any provisions for inserting varying percentage rates. Thus, it would be convenient to be able to write this rate directly into the problem definition.

Any value--or any group of symbols--which is to be used in the program without alteration is called a "constant." The user will find many uses for numeric constants such as the discount rate mentioned, for alphabetic constants, such as names and titles to be printed out on final reports, and for alphameric constants, which may serve any number of purposes.

In some circumstances, it will be convenient to write the constant directly in a decision table. In this case it will be called a "literal." In other cases, it will be more convenient to give the constant a name and store it within the system so that it can be called for by name when required. In this case it will be called a "named constant."

As an aid to the user, certain standard constants, such as the value 0 and the blank, have been "pre-named." These values are defined in the language vocabulary and they have already been given names. Thus, the programmer can write these names in the procedure statements without having to define them in the data description. These special constants, called "figurative constants," will be discussed later in this section.

Literals and named constants may be used in decision tables for the same purposes for which data-names are used--that is, as "operands." The essential difference between them is that a literal expresses an actual value--a value to be read "literally" at the point where it is written--whereas a named constant is the "name" of such a value, and it cannot be used, or interpreted, in a decision table until it has been defined in the data description.

The following example will show the difference between literals and named constants:

From FIGURE 2.2, page 2.2, the condition in line 1, rule 1 reads:

SPECIAL. CLEARANCE EQUALS 1

The condition indicates that the value for the data name SPECIAL. CLEARANCE equals 1. In this case the 1 is a literal.

The value 1 could be placed in the data description and give it a name "ONE." The condition is now stated as:

SPECIAL. CLEARANCE EQUALS ONE

The same result is obtained in either case, and it may appear at first sight that it is more efficient to write literals than named constants. This may or may not be so. If the constant is short, as in this example, it will usually be more convenient to write it as a literal. If it is long, and if it can be given a short name, it may be more efficient to treat it as a named constant.

LITERALS. Although a literal may be written and used in a decision table as if it were a data-name, it differs from data-names (including named constants) in that its value is the value literally stated--it is not used as a name for some other value.

RULES FOR FORMING LITERALS

Literals may be numeric, alphabetic, or alphameric. Some of the rules for forming numeric literals differ slightly from the rules for alphabetic and alphameric literals. For convenience of reference, the rules governing each type are listed separately.

NUMERIC LITERALS

1. All literals are limited to 10 characters in length.

2. Numeric literals may contain only numerals, not more than one decimal point, and a plus or minus sign to indicate whether the value of the number is positive or negative. "Floating point" numbers also contain the letter F, as explained in Rule 3 below, and may contain more than one plus or minus sign. The decimal point is required except where it would be the last character of the literal; in that case it must <u>not</u> be used. The decimal point will be noted by the system in order to align the number properly for use, and it is not counted in determining the length of the literal.

3. Numeric values may be entered as "floating point" numbers by writing the "fraction" (i.e., the number or decimal fraction), then

the symbol F, and then the exponent. The fraction and the exponent may each have a plus or minus sign. The symbol F. is not counted in determining the length of the literal. The system will accept floating point numbers using a base of 10 only. (A floating point number is a number expressed as a decimal number or decimal fraction multiplied by some power of 10. For example, the number 1500 might be written as 1.5F3, which is equivalent to 1.5 times 10^{3}., the same number might also be written as 15F2,.15F4, or in any other similar way that is convenient. The number .002, which is equivalent to 2 times 10^{-3}, might be written 2F-3.)

4. Numeric literals must not be enclosed in quotation marks.

ALPHABETIC AND ALPHAMERIC LITERALS

1. An alphabetic or alphameric literal may contain any of the characters from the basic character set except the quotation mark. A blank is treated as a character and may be included in an alphameric literal.

2. Like numeric literals, alphabetic and alphameric literals are limited to 10 characters in length.

3. All non-numeric literals must be enclosed in quotation marks to distinguish them from names. This rule applies even should the literal contain symbols (such as the arithmetic symbols and the blank) which may not be used in names.

FIGURATIVE CONSTANTS resemble named constants except that their names are already assigned values, so that the user need not write data description entries for them.

Figurative constants are names for certain constant quantities which are used frequently in data processing systems. The list includes names which represent zeros and blanks. Following is a list of the figurative constants:

ZERO or ZEROS or ZEROES

BLANKS or BLANKS

In general, a figurative constant is used to place the value it names in a given storage area, although it is not limited to this usage. For example, if the user wishes to reduce the value of a data item called COUNTER to zero, he can do so by writing the instructions MOVE ZEROS TO COUNTER. This procedure will replace all previous data in COUNTER by zeros. Similarly, if he wished to erase all data in an area called AMOUNT, he could write MOVE BLANKS TO AMOUNT. In each case, the specified area will be completely filled with characters of the value named.

FIGURATIVE CONDITION NAMES are similar to figurative constants and are used to facilitate the expression of conditions. They are used to test for "sign conditions," "class conditions" and "conditional expressions."

The FIGURATIVE SIGN CONDITION NAMES ARE:

POSITIVE
NEGATIVE
ZERO

The sign condition can be used only in conjunction with numeric data. This form of condition may be used to express a test to see if an item satisfies one of the following:

1. A "negative condition" (is less than zero).
2. A "zero" condition (equals zero).
3. A "positive condition" (is greater than zero).

The general form for writing the sign condition is:

operand	operator	operand
NAME	EQ	POSITIVE ZERO NEGATIVE

The FIGURATIVE CLASS CONDITION NAMES are:

NUMERIC
ALPHABETIC

The class conditions are the same as defined in Chapter 5.

CLASS	DESCRIPTION
NUMERIC	Consists entirely of digits (0-9); may also contain an operational plus or minus sign; an actual decimal point is a non-numeric character and is not permitted.
ALPHABETIC	Consists entirely of letters of the alphabet; may also contain one or more blanks.
ALPHAMERIC	Consists of any characters from the basic character set. May be wholly numeric or alphabetic.

The class conditions are used primarily for validating input data.

The general format for writing the class

operand	operator	operand
NAME	EQ	NUMERIC ALPHABETIC
NAME	NE	NUMERIC ALPHABETIC

VERBS specify actions. Verbs are used in conjunction with names to form commands. Specific rules governing verbs are in Chapter 4.

OPERATORS. Not all words that cause action are verbs. Consider the sentence IF A = B THEN MOVE A TO OUTPUT. This sentence contains only one verb, the word MOVE, yet it implies two separate operations. The MOVE operation, of course, is one of them; the other is a test to determine if the condition A=B has been met. However, the user does not have to write a verb directing a test for this condition. The language contains several words and symbols (such as the arithmetic symbols) which are not verbs but which cause operations. They are called "operators." It is necessary to distinguish between verbs and operators, since they are used in different ways.

In general, operators specify "actions" or "relationships" without actually expressing them in verb form. There are two basic kinds of operators: ARITHMETIC OPERATORS and RELATIONAL OPERATORS.

ARITHMETIC OPERATORS. The complete list of arithmetic operators is given below. They are used in forming arithmetic expressions, as explained later in this chapter.

Operator	Meaning
+	Addition
-	Subtraction
*	Multiplication
/	Division
**	Exponentiation

RELATIONAL OPERATORS. The need frequently arises to make tests in order to determine what should be done next. This language provides a number of "relational operators" which enable the user to express the tests he wants performed. For example, the statement IF SALARY = ZERO is built around the relationship implied by the equal sign, which is a relational operator.

A complete list of the seven relational operators follows:

Symbol	Meaning
EQ	Is equal to
NE	Is not equal to
LT	Is lesser than
LE	Is lesser than or equal to
GT	Is greater than
GE	Is greater than or equal to
CMP	Compared

Compared (CMP) is included in the list more as a convenience for the decision table format than as a true relational operator.

Line	Operand 1	OP	Operand 2	Rule 1		Rule 2	
				OP	Operand 2	OP	Operand 2
1	DETAIL	CMP	MASTER	EQ		GT	
2	DETAIL			EQ	MASTER	GT	MASTER
3	TRANS. COD	EQ			ADDITION		CHANGE

This convenience is demonstrated by examining the conditions represented in lines 1 and 2 above. Each line represents the same conditions: Detail is equal to master (Rule 1) and detail is greater than master (Rule 2). In line 1, the master is written only once, while in line 2 it is written twice. This allows both operands to be placed in the stub and the relational operator placed in the entry. In line 3, operand 1 (trans. code) and the operator (eq) are factored into the stub and the different operand 2's placed in the entry.

Relational operators are combined with data names, literals, etc. to form conditional expressions. The detailed rules for using the relational operators are explained later under Conditional Expressions.

EXPRESSIONS. An expression may be defined as a meaningful combination of names, literals, and/or operators which may be reduced to a single value. This definition will become clear after the reader has studied the two types of expressions - the "arithmetic" expression and the "conditional" expression.

Since arithmetic expressions do not fit into the decision table conveniently, they must be defined and named in the data description and the expression name is used in the decision table. Each time an arithmetic expression is called for in a condition or action, its value is determined anew. The value of an expression does not persist.

ARITHMETIC EXPRESSION is a combination of data-names and numeric literals joined by one or more arithmetic operators in such a way that the entire expression can be reduced to a single numeric value. Special cases of arithmetic expressions are combined with the data movement verbs SET and MOVE which are explained in Chapter 4.

The following are examples of arithmetic expressions:

```
NET. PAY = (HOURS + OVERTIME * 1.5)  * WAGE. RATE - FICA

VOL = PI * RADIUS ** 2 * HEIGHT / 3

SALES. C = WEEKLY. SALES * .05
```

Note that each of the above expressions is a combination of data-names and/or literals joined by arithmetic operators. At object time, each data-name will

represent a value and, in each of the above examples, one numeric value will result from the specified computation. Thus, if WEEKLY.SALES has the value 574.20, the SALES.C would reduce to the value of 28.71.

ORDER OF COMPUTATION IN ARITHMETIC EXPRESSIONS. The way in which an arithmetic expression is to be evaluated can be specified by parentheses. Thus, the expression D = A * B + C might be considered ambiguous. Does the user mean D = (A * B) + C, or does he mean D = A * (B + C)? The user may use pairs of parentheses in order to describe exactly the way in which he wants the computation to proceed.

If parentheses are not written to specify the order of computation, the arithmetic expression is evaluated using the following rules:

1. All exponentiation is performed first.

2. Then, multiplication and division are performed.

3. Finally, addition and subtraction are performed.

4. In each of the three above steps, computation starts at the left of the expression and proceeds to the right. Thus, A * B / C is computed as (A * B) / C, and A / B * C is computed as (A / B) * C.

5. When parentheses are present, computation begins with the innermost set and proceeds to the outermost. Items grouped in parentheses will be evaluated in accordance with the above rules, and the result will then be treated as if the parentheses were removed.

A CONDITIONAL EXPRESSION is an expression which, taken as a whole, may be either true or false, depending on conditions existing when the expression is examined. Generally, a conditional expression contains at least one variable quantity, and the truth or falsity of the expression will depend on the particular value assumed by the variable or variables. For example, the expression A IS GREATER THAN 10 is conditional, since it may or may not be true, depending on the value of the quantity A. Obviously, if A had a value of 12, the expression would be true; if the value were 7, the expression would be false.

A conditional expression may contain data-names, condition-names, names of arithmetic expressions, and expressions which show relationships between values (such as the expression IS GREATER THAN).

Conditional expressions are always written in the decision table; never in the data description. This is one of the major advantages of decision tables - they offer a two dimensional visual aid in exploring the potential combinations of conditions. Conditional expressions are discussed here in three categories:

1. Simple conditional expression.

2. "AND" compound conditional expression - simple conditional expressions joined by "and's."

3. "OR" compound conditional expressions - conditional expressions of category 1 and/or 2 joined by "or's."

Conditional expressions exist as that part of a data rule which must be satisfied prior to taking the indicated actions of that same data rule.

The three categories can best be explained with an illustration.

					Rule 1		Rule 2		Rule 3	
Line	Verb	Operand 1	Op	Operand2	Op	Operand2	Op	Operand 2	Op	Operand 2
1		MAR. STATUS	EQ			MARRIED		DIVORCED		SINGLE
2		AGE			GT	21	GT	30		
3		AGE	LT	39		Y		Y		
4		HRLY. RATE	GT			3.50		4.00		2.00
5		HRLY. RATE	LT	5.00		Y		Y		Y
6	DO	STATISTICS				X		X		X
7	GO TO	CONTINUE				X		X		X

A SIMPLE CONDITIONAL EXPRESSION (category 1) is composed of two operands connected by a relational operator and is illustrated by 10 different conditions in the above example. They are:

Line 1 — marital status is married
marital status is divorced
marital status is single

Line 2 — age is greater than 21
age is greater than 30

Line 3 — age is less than 39

Line 4 — hourly rate is greater than 3.50
hourly rate is greater than 4.00
hourly rate is greater than 2.00

Line 5 — hourly rate is less than 5.00

An "AND" COMPOUND CONDITIONAL CONDITION is illustrated by the interpretation of Rule 1. If marital status is married and age is greater than 21 and age is less than 39 and hourly rate is greater than 3.50 and hourly rate is less than 5.00...this portion of Rule 1 shows 5 simple conditional expressions connected by four "AND'S." Similar interpretations of Rule 2 and 3 can be made.

An "OR" COMPOUND CONDITIONAL EXPRESSION is never shown in one rule in a decision table. We can see that all three of the above rules have the same actions. Therefore, we have expressed the "OR" conditions for the same actions as separate rules. All three rules could be stated in narrative in a single sentence as follows. If (marital status is married and age is greater than 21 and age is less than 39 and hourly rate is greater than 3.50 and hourly rate is less than 5.00) or (marital status is divorced and age is greater than 30 and age is less than 39 and hourly rate is greater than 4.00 and hourly rate is less than 5.00) or (marital status is single and hourly rate is greater than 2.00 and hourly rate is less than 5.00) then do statistics routine and continue to next step.

We can readily see that the tabular display of the condition combinations has much more human communication value.

In forming the simple conditional expression (operand 1, operator, operand 2), operand 1 is always written in the left operand column in the table stub. The relational operator may be written in the stub or the entry. If the operator is common to all rules for this line it may be factored into the stub, see lines 1, 3, 4 and 5 above. If it varies from one rule to another then it must be written in the entry, see line 2 above. Operand 2 may be written in the right operand of the table stub if it is common to all rules for this line, see lines 3 and 5 above. Operand 2 must be written in the table entry if it varies from one rule to another, see lines 1, 2 and 4 above.

The following table shows the permissible combinations of name-types for condition operands 1 and 2:

Oper-and 1 \ Oper-and 2	Data Name	Condition Name	Literal	Expression Name	Named Constant	Figurative Sign Condition Names	Figurative Class Condition Names	Figurative Constant Names
Data Name	Y	Y	Y	Y	Y	Y	Y	Y
Literal	Y	N	N	Y	N	N	N	N
Expression Name	Y	Y	Y	Y	Y	Y	N	Y
Named Constants	Y	N	N	Y	N	N	N	N

Y= Permissible

N= Not Permissible

REPORT GENERATION FEATURES are included in this language to accomplish such functions as:

Print Headings per page and per change in control fields data.

Automatically increment and print page number.

Print date on each page of report.

Group suppress printing of control field data.

Print selected field totals automatically per change in control field data and at end of report.

Generally, edit fields in lines of print such as suppress leading zeros, insert periods and commas in quantity fields, floating dollar signs and asterisk check protection.

These functions are accomplished by data descriptions. The main flow of the problem definition prepares a working storage of unedited data for a detail line. The command DO REPT n (where n is the report number) assumes that the detail line is edited for print in the manner specified by the data description including the consideration of all the other report functions defined in the data description. At the end of the report the command DO END.REPT n is given to effect the end of report functions such as printing all required levels of totals (including report total).

Chapter 4: DECISION TABLE FORMAT AND VERBS

INTRODUCTION This chapter is concerned with the current verbs, commands, format and sequence control of decision tables. Refer to Chapter 2 for description and terminology of general concepts, structure and elements of decision tables.

VERBS The list of verbs is given below:

READ WRITE	Input = Output
MOVE SET	Data Transmission
MOVE SET	Arithmetic
GO TO NEXT ELSE DO STOP	Table Sequence Control

This language is designed to be "open-ended." That is, the list of verbs is never closed. For instance, it is known at this time that verbs should be included for "data table operations" such as "search," "delete" and "insert," but time does not permit this inclusion.

COMMANDS call for action(s) to be executed. Each verb in the preceding list forms the basis for a specific command. A command normally consists of a verb followed by operand 1, an operator and operand 2.

FORMAT OF DECISION TABLE The decision portion of a problem definition is written on the Decision Table Form designed for that purpose. The form is shown on Page 4.2.

TABLE TITLE Indicative title of decision table contents.

SYSTEM SEGMENT A logical subdivision of the system being defined. It may be a functional subdivision such as "material control," "credit clearing," etc., or further subdivisions. It may be computer runs, EAM processing or a clerical activity.

SYSTEM A large functional or goal-oriented activity which is being defined. Examples of functional activities (usually thought of as vertical slices of an organization) are accounting, industrial relations, engineering, manufacturing, marketing. Examples of goal-oriented activities (usually thought of as horizontal slices of an organization which cross and absorb parts of many vertical

TABLE TITLE	SYSTEM SEGMENT	SYSTEM	PREPARED BY _____	OPEN ___ CLOSED__	DATE	ELSE	NEXT TABLE

LINE NR	VERB	OPERAND 1	OP	OPERAND2	RULE	FREQ	RULE	FREQ	RULE	FREQ	RULE	FREQ	RULE	FREQ	RULE	FREQ
					*	OPERAND2	*	OPERAND2	*	OPERAND2	*	OPERAND2	*	OPERAND2	*	OPERAND2

*For conditions: OPERATOR
For actions: SEQUENCE

FIGURE 4.1

TABLE

activities) are design, manufacture, marketing, etc. of specific products or product groups, such as electrical transformers (size range), glass products, canned foods, matches.

PREPARED BY Initials of analyst.

OPEN _____ A check mark is entered if the table is in-line (referred to in other tables by the verb GOTO).

CLOSED _____ A check mark is entered if the table is out of line (referred to in other tables by the verb DO).

DATE Current date.

ELSE (TABLE) An entry may or may not be made. If no set of conditions for any rule is satisfied, a table number or name is entered; thus directing the consideration of the next table when no commands of the current table have been executed.

NEXT (TABLE) An entry may or may not be made. An entry is made when many or all of the rules GOTO the same next table. This eliminates the need of writing the GOTO TABLE.NAME in each of the rules. However, if a GOTO entry is made in a rule, it over-rides an entry the header NEXT.

TABLE (NAME OR NUMBER) A short identification of the table using either a name or number. The beginning table of a problem definition is identified by BEGIN if a name is used or by 001 if a number is used. This field appears in two places on the form for convenience of referencing, depending upon the manner of binding the definition material.

LINE NR Enter a different line number for each line used within a TABLE IDENTIFICATION.

VERB This column is used only for the action part of the table. Specific permissible verb entries are shown in the above list and in the expanded discussion later in this chapter.

OPERAND 1 An entry is always made in this column for each conditional expression and command. Permissible entries for conditional expression are shown in Chapter 3, Permissible entries in operand 1 for commands are defined with each verb discussion later in this chapter.

OP (OPERATOR) For conditional expressions, the permissible entries are the seven relational operators discussed on Page 3.12. For commands, the permissible entries are described, with their associated verbs, later in this chapter.

OPERAND 2 Heading appears in both the "stub" and "entry" parts of the table. If operand 2 data is the same for all rules affected by the current

line, then the operand 2 data may be entered in the "stub" and a "limited entry" is made in the "entry" part of the table. If operand 2 data varies from rule to rule for the current line, then operand 2 data must be written in the "entry" part of the table. For conditional expressions, the permissible entries in operand 2 are shown in Chapter 3, Permissible entries in operand 2 for commands are defined with each verb discussion later in this chapter.

RULE (NUMBER) An identity number given to each decision rule within a table name (or number) for reference.

FREQ (Frequency) An entry is made showing the number of times the action part of a rule will be executed for a specific time interval. The time interval should be specified as a constant for each problem definition; i.e., weekly, daily, hourly, monthly, etc. If the frequency varies greatly, "note numbers" can be entered and under the "note section" the frequency can be expanded to reflect peaks and valleys in relation to time periods. Frequency data can be used to assist in determining the best processing medium and equipment requirements.

* This is a dual purpose column. For conditional expressions, it is used to insert relational operators (similar to the "operator" column in the "stub"). For actions (commands), it is used to specify only the mandatory sequence in which the actions must be executed. The method of numbering the actions is such that a maximum degree of flexibility is permitted in implementing the action sequence in a step by step procedure.

					Rule 1		Rule 2		Rule 3	
Line	Verb	Operand 1	OP	Operand 2	*	Operand2	*	Operand2	*	Operand2
10	Read	File 1			1	X	1	X		X
11	Write	File 2			1	X	3	X		X
12	Move	RCD 1	To	RCD 2			2	X		X
13	Set	FLD 1	EQ	FLD 2						X
14	Do	TAB 006			2	X	3	X		X
15	GOTO	TAB 020			3	X	4	X		X

Figure 4.2

Sequencing actions per rule will be explained by discussing the above example.

RULE 1 The ones entered in lines 10 and 11 indicate that the action on either line 10 or 11 can be taken first. The 2 in line 14 indicates that the action on line 14 must be taken after the actions on line 10 and 11 are executed. Likewise, the 3 in line 15 indicates the action in line 15 must be taken after the action in line 14. Summarizing, we can see that there are two possible sequences of executing actions for rule 1: (1) lines 10, 11, 14, 15 or (2) lines 11, 10, 14, 15.

RULE 2 Following similar logic for Rule 1, we find that there are two possible sequences of executing actions for Rule 2: (1) Lines 10, 12, 11, 14, 15 or (2) Lines 10, 12, 14, 11, 15.

RULE 3 When no data is entered for action sequence, it means that the actions must be executed in the sequence written.

INPUT/OUTPUT COMMANDS In considering any level of data processing problem definition, it is convenient to think of it in terms of inputs, reference files, processing decision logic and outputs. These commands are involved with the reading of records from input and reference files and writing records on output and reference files. Since the language, at this time, is not involved with providing for its mechanized implementation, such verbs as "open" and "close" files are not included.

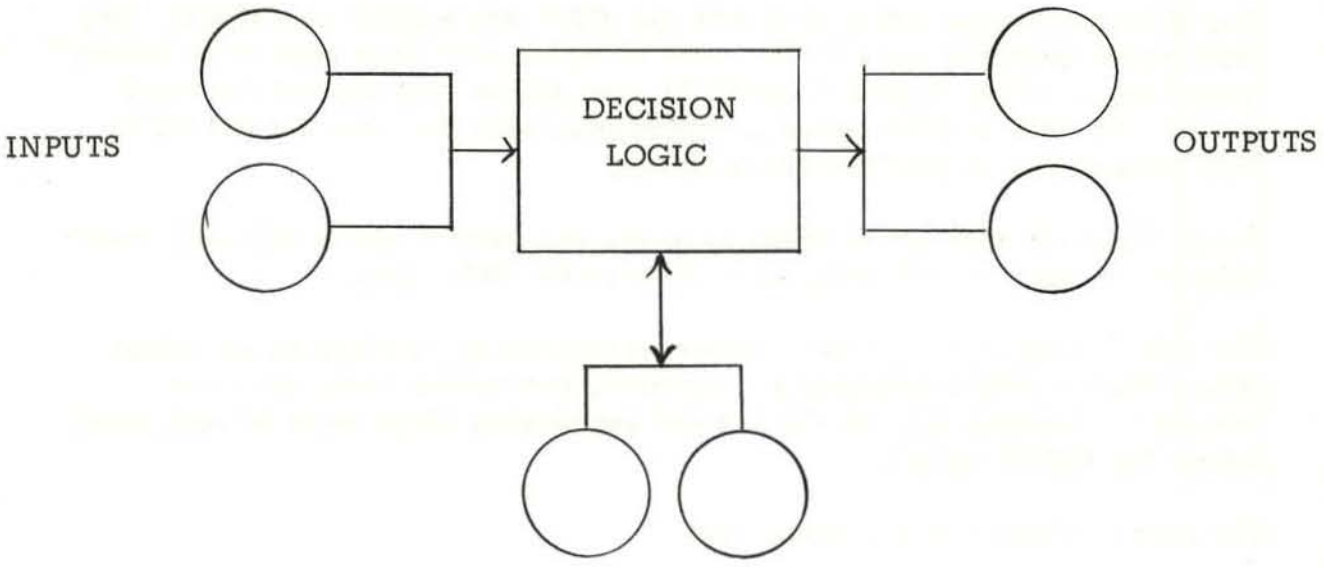


Figure 4.3

The READ command is used to fetch a record from an input or reference file and make it available for processing. When a READ is executed, the next record in the named file becomes accessible in the input area defined by the associated Record Description in the Data Description. The record remains available in the input area until the next READ (for that file) is executed.

If a file contains more than one type of record, the READ verb delivers the next record regardless of type. The differing records automatically share the same input area; thus the user must provide for determining the type of the current record and must refer only to information that is present in the current record.

The format of the READ command is:

Verb	Operand 1	OP	Operand 2	Rule 1 Operand 2	Rule 2 Operand 2	Rule 3 Operand 2
READ	RCD. NAME					
READ	RCD. NAME	INTO	AREA. NAME			
READ	RCD. NAME	INTO		AREA. NAME 1	AREA. NAME 2	AREA. NAME 3

Figure 4.4

The INTO area-name option converts the READ into a READ and MOVE. The area-name specified must be the name of either a working area or an output record area. If the format of the INTO area differs from that of the input record, the data will be moved in accordance with the rules for the MOVE verb without the CORRESPONDING option.

When the INTO area-name option is used, the current record becomes available in the input record area, as well as in the INTO area.

The WRITE command is used to release a record for insertion in an output file. When a WRITE command is executed, the record-name record is released. Accordingly, all the desired processing steps must be performed before the WRITE occurs.

The format of the WRITE command is:

Verb	Operand 1	OP	Operand 2	Rule 1 Operand 2	Rule 2 Operand 2	Rule 3 Operand 2
WRITE	RCD. NAME					
WRITE	RCD. NAME	FROM	AREA. NAME			
WRITE	RCD. NAME	FROM		AREA. NAME 1	AREA. NAME 2	AREA. NAME 3

Figure 4.5

The FROM "area-name" option of the WRITE command is comparable to the INTO area-name option of the READ command. It, effectively, converts the WRITE into a MOVE and WRITE. The "area-name" must be the name of an input record area, a working area, or a constant area. If the format of the FROM area differs from that of "record-name," the data will be moved in accordance with the rules for the MOVE verb without the CORRESPONDING oprion. When the FROM "area-name" option is used, the information in the "area-name" area continues to be available.

Note: The names used for "record-name" and for "area-name" cannot be the same.

DATA TRANSMISSION COMMANDS The two data transmission verbs are MOVE and SET. Both verbs are involved with the transmission of data from one area to another. Differences in SET and MOVE are shown by the reversal of the source and receiving areas.

Verb	Source Area	Receive Area
SET	Operand 2	Operand 1
MOVE	Operand 1	Operand 2

Entries in the "source area" may be:

- A variable name
- A constant name
- A condition na me
- An expression name
- A figurative constant (blank or zero)
- A record name
- A literal

Entries in the "receiving area" may be:

- A variable name
- A record name

The MOVE command is used to transfer data from one data area (operand 1) to another area (operand 2). When a move command (1) is common to several rules of a table and (2) has the same data (operand 1) to be moved into a different data area (operand 2) for each rule, then this can be accomplished on one line. The format of the MOVE command is:

Line	Verb	Operand 1	OP	Operand 2	Rule 1 Operand 2	Rule 2 Operand 2	Rule 3 Operand 2
10	MOVE	DATA. NAME 1 OR LITERAL	TO	DATA. NAME 2			
11	MOVE	DATA. NAME 1 OR LITERAL	TO		DATA. NAME 2	DATA. NAME 3	DATA. NAME4
12	MOVE C	DATA. NAME1	TO	DATA. NAME 2			
13	MOVE C	DATA. NAME1	TO		DATA. NAME2	DATA. NAME3	DATA. NAME4

Figure 4.6

The following rules must be observed in writing MOVE commands:

1. Information from numeric fields may be moved to other numeric fields, to alphameric fields, and to report fields.

2. Information from alphabetic or alphameric fields may be moved only to other alphabetic or alphameric fields.

When the simple MOVE (lines 10 and 11 above) is executed, the data represented by DATA. NAME 1 or the specified literal is moved to the area designated by AREA. NAME 2, 3 or 4. This movement does not destroy the original data - it makes "copies" of it in the designated areas.

When both the source and the receiving areas are elementary items, editing appropriate to the format of the receiving area occurs automatically in the execution of the MOVE. The editing that is performed depends on whether the source data (specified by data-name-1 or literal) is numeric or non-numeric, as follows:

NUMERIC DATA ITEMS

1. The data from the source area is aligned with respect to the decimal point (assumed or actual) in the receiving area. This alignment may result in the loss of leading digits or of low-order digits (or both if the source area is larger than the receiving area).

2. If required by the format of the receiving area, zeros are replaced by spaces (blanks); and dollar signs, decimal points, and commas are inserted.

3. If no decimal point has been specified, the data will be right justified unless the data description of the item specifies JUSTIFIED LEFT.

NON-NUMERIC DATA ITEMS

1. The data from the source area is placed in the receiving area beginning at the left, unless the data description of the receiving area specifies JUSTIFIED RIGHT. Note that when a group item is moved, left justification is standard.

2. If the receiving area is not completely filled by the data being moved, the remaining positions are filled with spaces.

3. If the receiving area cannot contain all of the data being transferred, the MOVE terminates when the receiving area is filled.

FIGURE 4.7 contains several examples illustrating the editing feature of the MOVE verb.

SOURCE AREA			RECEIVING AREA		
Picture	Data before MOVE	Data after MOVE	Picture	Data before MOVE	Data after MOVE
99V99	1234	1234	99V99	9876	1234
99V99	1234	1234	99V9	987	123
99V99	1236	1236	99V9	987	124
9V9	12	12	99V999	98765	01200
XXX	A2B	A2B	XXXXX	Y9X8W	A2B
9V99	123	123	99.99	87.65	01.23
AAAAAA	REPORT	REPORT	AAA	JKL	REP
99V99	1234	1234	$ZZZ9.99	$8765.43	$ 12.34

Figure 4.7
Examples of data before and after MOVE is executed. Standard justification is assumed

Note that in each case in Figure 4.7 the data in the source area remains unaltered after the MOVE has been executed. Note also, as in the fourth example, that the information in any excess positions of a non-numeric receiving area is replaced by spaces at the right.

The corresponding MOVEC option (Figure 4.6, lines 12 and 13) of the MOVE command permits the user to specify the transfer of a group item containing one or more elementary items that require editing in conjunction with the MOVE. When a MOVEC command is executed, selected items within the source area (data-name-1 area) are moved, with any required editing, to selected areas within the receiving area (data-name-2, data-name-3, etc.). Items are selected by matching the data-names of items within data-name-1 with likedata-names of areas within data-name-2, according to these rules:

1. At least one of the items of a selected pair must be an elementary item.

2. The two data-names must be identical, including all qualification up to but not including data-name-1 and data-name 2.

Each corresponding item in the source area is moved to its corresponding receiving area. Editing appropriate to the format of the receiving area takes place automatically. The rules stated for the simple MOVE apply to each pair of corresponding items in the MOVEC; thus, the effect of a MOVEC command statement is equivalent to a series of simple MOVE commands.

To illustrate the use of MOVEC, suppose that the user wishes to transfer corresponding items from a work area named INV. POSTING to an output area designated INV. RCD.

Verb	Operand 1	OP	Operand 2
MOVEC	INV. POSTING	TO	INV. RCD

Figure 4.8

FIGURE 4.9 shows the movement of data that might result from this statement. Note that non-corresponding items in the source area are not moved and that non-corresponding items in the receiving area are not affected.

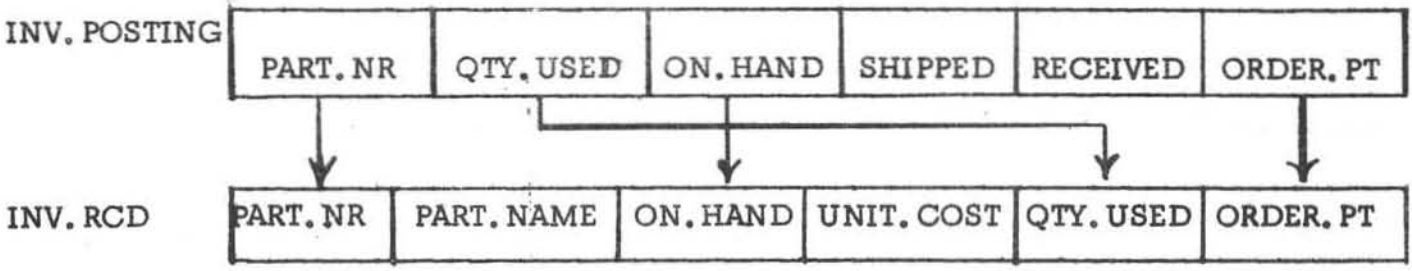


Figure 4.9
Movement of data resulting from execution of MOVEC

The SET command is used to transfer data from one data area (operand 2) to another data area (operand 1).

The SET command has unique utility in conjunction with decision tables in that, on one line, a single "receive area" can obtain data from one of several "source areas." The format of the SET command is:

Line	Verb	Operand 1	OP	Operand 2	Rule 1 Operand 2	Rule 2 Operand 2	Rule 3 Operand 2
10	SET	DATA. NAME 1	EQ	DATA. NAME 2 LITERAL			
11	SET	DATA. NAME 1	EQ		DATA. NAME 2 LITERAL	DATA. NAME3 LITERAL	DATA. NAME4 LITERAL
12	SETC	DATA. NAME 1	EQ	DATA. NAME 2			
13	SETC	DATA. NAME 1	EQ		DATA. NAME2	DATA. NAME3	DATA. NAME4

Figure 4.10

Lines 10 and 11 are the simple SET command and have the same rules of data transmission as the simple MOVE command. Lines 12 and 13 are the SETC (set corresponding) commands and have the same rules of data transmission as the MOVEC command explained above.

ARITHMETIC COMMANDS There are two direct arithmetic commands MOVE and SET. For convenience of the decision table format, the MOVE and SET command functions are expanded to include arithmetic (addition, subtraction, division and multiplication). An indirect method of performing arithmetic operations is via the name of an arithmetic expression. Figure 4.11 shows several examples of how arithmetic operations can be accomplished.

The rules for numeric data transmission apply to the SET and MOVE arithmetic commands. When using the SETC and MOVEC commands the DATA. NAMES must represent areas of storage which are composed of smaller units of fields. The effect of the MOVEC DATA. NAME. 1 to + DATA. NAME. 2 is to cause each numeric field in DATA. NAME. 1 to be added to its corresponding field (i.e. a field with the same name) in the DATA. NAME. 2. Non-corresponding and non-numeric fields in DATA. NAME. 1 and in DATA. NAME. 2 are not affected.

Intermediate results are carried with an extra digit to both the right and left of the receiving area field. Intermediate results are automatically truncated or rounded to the number of places indicated by the format description of the receiving field.

TABLE TITLE	SYSTEM SEGMENT	SYSTEM	PREPARED BY	OPEN CLOSED	DATE	ELSE	NEXT TABLE

LINE NR	VERB	OPERAND 1	OP	OPERAND2	RULE 1	FREQ	RULE 2	FREQ	RULE 3	FREQ	RULE	FREQ	RULE	FREQ	RULE	FREQ
					*	OPERAND2	*	OPERAND2	*	OPERAND2	*	OPERAND2	*	OPERAND2	*	OPERAND2
10	SET	DATA.NAME1	EQ	DATA.NAME2												
11	SET	DATA.NAME1	EQ			DATA.NAME2		DATA.NAME3		DATA.NAME4						
12	SETC	DATA.NAME1	EQ	DATA.NAME2												
13	SETC	DATA.NAME1	EQ			DATA.NAME2		DATA.NAME3		DATA.NAME4						
14	MOVE	DATA.NAME1	TO	DATA.NAME2												
15	MOVE	DATA.NAME1	TO			DATA.NAME2		DATA.NAME3		DATA.NAME.4						
16	MOVEC	DATA.NAME1	TO	DATA.NAME2												
17	MOVEC	DATA.NAME1	TO			DATA.NAME2		DATA.NAME3		DATA.NAME4						
18	SET	DATA.NAME	EQ	EXP.NAME1												
19	SET	DATA.NAME	EQ	EXP.NAME1												
20	SET	DATA.NAME	EQ			EXP.NAME1		EXP.NAME2		EXP.NAME3						
21	MOVE	EXP.NAME	TO	DATA.NAME1												
22	MOVE	EXP.NAME	TO	DATA.NAME1												
23	MOVE	EXP.NAME	TO			DATA.NAME1		DATA.NAME2		DATA.NAME3						

*For conditions: OPERATOR
For actions: SEQUENCE

FIGURE 4.11

TABLE

The SET command format is:

SET DATA.NAME 1 EQ A DATA.NAME 2
EXP.NAME

where "A" is (may be blank when operand 2 is an expression name), an arithmetic operator +, -, / or x. Examples are shown in Figure 4.11, lines 10, 11, 12, 13, 18, 19 and 20. Operand 1 (Data.Name 1) is the receiving area and operand 2 (Data.Name 2 or Exp. Name) is the source area.

Receive Area + Source Area → Receive Area
Receive Area - Source Area → Receive Area
Receive Area / Source Area → Receive Area
Receive Area x Source Area → Receive Area

The MOVE command format is

MOVE DATA.NAME 1 To A DATA.NAME 2
EXP.NAME

where "A" is (may be blank when operand 1 is an expression name) an arithmetic operator +, -, /, or x. Examples are shown in Figure 4.11 - lines 14, 15, 16, 17, 21, 22 and 23. Operand 1 (Data.Name 1 or Exp. Name) is the source area and operand 2 (Data.Name 2) is the receiving area.

Receive Area + Source → Receive Area
Receive Area / Source → Receive Area
Receive Area - Source → Receive Area
Receive Area x Source → Receive Area

GENERAL RULES FOR ARITHMETIC COMMANDS

1. All fields involved must be numeric. Constants cannot be the receiving area.

2. The format of the operands may be different. Decimal point alignment is automatically supplied throughout the computation.

3. The format of any item involved in computations cannot contain editing symbols.

4. Valid combinations of the "source area" and "receiving area" are:

Source Area \ Receiving Area	Variable Name	Record Name
EXPRESSION NAME	Y	N
VARIABLE NAME	Y	N
CONSTANT NAME	Y	N
LITERAL	Y	N
RECORD NAME	N	Y

5. An arithmetic expression is computed and the results are used as the source. The expression is recomputed each time it is used in a command. Its value does not persist.

FIGURE 4.12 shows examples to clarify these rules:

SOURCE AREA			Operator	RECEIVING AREA		
Picture	Data Before Move Or Set	Data After Move Or Set		Picture	Data Before Move Or Set	Data After Move Or Set
99V99	0110	0110	+	99V99	2120	2230
99V99	1013	1013	-	99V9	333	232
9V99	111	111	/	99V9	990	892
9V9	55	55	X	99V99	1111	6111

Figure 4.12

TABLE SEQUENCE CONTROL COMMANDS Procedural sequence control is accomplished, primarily, via table sequence control verbs, GOTO, DO, ELSE, NEXT and STOP. (It is accomplished to a lesser degree within a rule by the action, or command, sequence entries.)

The GOTO COMMAND appears in the body of the decision table and directs the procedure to the next TABLE.NAME to be considered. The next TABLE.NAME may be the same as the TABLE.NAME in the current table. . . The GOTO command is always the last action of every rule. If the GOTO COMMAND does not appear in a rule, then the next TABLE.NAME is taken from the table header NEXT. A GOTO COMMAND will over-ride the NEXT COMMAND in the header.

The ELSE COMMAND also appears in the header and directs the procedure to consider the next TABLE.NAME whenever no set of conditions for any rule is satisfied.

The DO COMMAND provides a means of interrupting the actions in a rule of the current table in order to consider another table. After the other table is considered (and actions taken) control comes back to the next action of the rule for the current table.

Special operands of the DO COMMAND are used to specify the execution of report generation functions(RGF). The three operands are REPT.n, BEG.REPT.n and END.REPT.n where "n" is a report number assigned in the data description. The report generation features (DO REPT.n) are discussed in Chapters 3 and 5. The DO BEG.REPT.n indicates to the RGF that it is to process the beginning of the report functions such as: write heading lines. The DO END.REPT.n indicates to the RGF that it has processed the last detail record for the report and commands it to accomplish the end of report functions such as printing report totals. There are no tables written (by the person defining the problem) with the names REPT.n, BEG.REPT.n and END.REPT.n. It is assumed that the report data description, report planning sheet and the sample of the report in conjunction with the report commands are sufficient specifications for a programmer or procedure writer to implement the report.

TABLE TITLE	SYSTEM SEGMENT	SYSTEM	PREPARED	OPEN X	DATE	ELSE	NEXT	TABLE
PROCESS TRANSACTIONS		INVENTORY	BY OYE	CLOSED	8/30/61			TRAN

LINE NR	VERB	OPERAND 1	OP	OPERAND2	RULE 1 FREQ		RULE 2 FREQ		RULE 3 FREQ		RULE 4 FREQ		RULE FREQ		RULE FREQ	
					*	OPERAND2	*	OPERAND2	*	OPERAND2	*	OPERAND2	*	OPERAND2	*	OPERAND2
1		TRANS. CODE	EQ			ISSUE		RECEIPT		ORDER		ELSE				
10	SET	INV BAL	EQ-		1	ISSUE QTY										
11	"	" "	EQ+				1	RCPT.QTY								
12	MOVE	INV BAL	TO	PRNT BAL	2	X	2	X								
13	SET	ISSUE.YTD	EQ+	ISSUE QTY	1	X										
14	SET	RCPT.YTD	EQ+	RCPT QTY			1	X								
15	DO								1	ORD. RTNE	1	ERROR				
16	GOTO	FILE.MATCH			3	X	3	X	2	X	2	X				

*For conditions: OPERATOR
For actions: SEQUENCE

FIGURE 4.13

TABLE

The STOP COMMAND is used to specify a temporary or final halt. Its general format is:

Verb	Operand 1	Op	Operand 2	Operand 2	Operand 2	Operand 2
STOP	LITERAL					
STOP				LITERAL 1	LITERAL 2	LITERAL 3

The literal is numeric. The STOP COMMAND can appear anywhere in the action sequence of a rule.

KINDS OF DECISION TABLES There are two major kinds of decision tables - open table and closed table.

The OPEN TABLE has these unique characteristics:

- It may be entered only by a GOTO COMMAND (the only exception is the BEGIN table).
- It may not be entered by a DO COMMAND.
- It may contain DO COMMAND(S).
- It will indicate the next table to be considered by a GOTO, NEXT or ELSE COMMAND.

The CLOSED TABLE has these unique characteristics:

- It may be entered only by a DO COMMAND.
- It may not be entered by a GOTO, NEXT or ELSE COMMAND.
- It may not contain a GOTO, NEXT or ELSE COMMAND.
- It may contain DO COMMANDS for tables other than itself.
- After having been considered (and actions taken), the sequence control is transferred back to the next action of the rule in that table from which control was transferred to this closed table.

MISCELLANEOUS CHARACTERISTICS OF TABLES AND RULES

UNCONDITIONAL TABLE
A table may contain only actions (commands) and no conditions (there would be only one data rule). This is called an UNCONDITIONAL table and would be of the closed or open kind of table.

NO ACTION RULE A rule may contain only condition(s) and the only action is the next table.

TABLE ENTITY A decision table is always considered as an entity. A transfer (via GOTO, DO, ELSE and NEXT) is always to a table as a whole - not to a rule or condition within a table.

ONE RULE SUCCESS PER TABLE For any one pass through a table only a single set of conditions for one rule can be satisfied and, therefore, only a single set of actions for the same rule can be executed. The rule that is satisfied may be the ELSE condition and the only action may be to go to the next table via the ELSE NEXT.TABLE.

RULE INDEPENDENCE Each rule is stated as an entity separate from other rules in the same table. A rule must contain sufficient conditions to insure the execution of its action(s) when the conditions are satisfied regardless of the sequence of considering rules in a table. This means that (1) all conditions pertinent to a rule are indicated and (2) conversely, conditions that would automatically be met in a current rule due to the failure of a set of conditions in a prior rule of the same table must still be indicated. The only exception to this RULE INDEPENDENCE is when an action is required for the ELSE condition other than "go to the next table." See discussion of ELSE below.

CONDITION INDEPENDENCE For a rule, the sequence of testing conditions is not relevant since all indicated conditions must be satisfied before the actions are executed. This is true from the viewpoint of defining a problem at a level as independent of procedural steps as possible. When the problem is implemented by a programmer or procedure writer, the sequence of testing conditions may have a significant effect on the efficiency of the process media.

TABLE FORM SIZE is designed to the usual 8-1/2 X 11 inches. Our eyes are trained for this size paper. It is believed that we can comprehend decision rules more quickly when this size form is used. The "one rule success per table" concept tends to keep the decision rules confined to this size form.

The ELSE condition rule is possible because of the "one rule success per table" concept. The entries in the table header ELSE may be:

- a TABLE.NAME
- left blank
- STOP.n COMMAND

This field is generally used for error detection but may be a systems logical sequence control. When an action is required for the ELSE other than "go to the next table," the word ELSE can be entered as a condition in operand 2 in the ENTRY portion of the table and the necessary actions indicated, see Rule 4 of FIGURE 4.13, page 4.17. In this case the ELSE rule has to be last rule considered for the table.

Chapter 5: DATA DESCRIPTION

PURPOSE A major purpose of writing a "data description" is to furnish the processing system with a means of identifying each item of data which has been named in the decision tables. This description must include all items on which the system is to operate.

Actually, the system needs more than a means of merely identifying the data. If the data is numeric, for instance, there must be a means of showing where the decimal point is located. If the system is to print out monetary values, it must have information on where to place the dollar sign, and whether or not to print leading zeros. Details of this sort are usually referred to as "editing." The actual commands for accomplishing details of this sort need not be stated explicitly. It is assumed that sufficient intelligence exists in the implementation to accomplish the implied transformations.

Writing a data description is not difficult, once a few basic principles are understood, and these are explained in the following pages. The reader should realize, moreover, that this method of describing data in a separate section of the program has an important advantage. The typical problem definition will deal repeatedly with data of the same kind. Use of a separate data description permits the user to describe each kind of data once, instead of having to describe each individual item as it occurs.

FILES, RECORDS AND FIELDS Before proceeding with the details of the data description it is only necessary to define three terms as they are used in this system: "file," "record," and "element." The exact implication of these terms varies from person to person and company to company.

A FILE is a body of data stored in some external medium which can be made accessible to the system. In this sense, an external medium is any medium which provides input to the system from without. Data in such a medium cannot be used by the system until it has been brought into it. The usual external medium of electronic data processing systems is magnetic tape and random access storage.

The concept of a file as "external" to the system carries certain implications which should be considered. These arise, not from the definition of a file, but rather from certain practical considerations.

For example, a file is usually a relatively large body of data, although there is no implied relationship between the length of a file and the storage capacity of a tape. Thus, there may be more than one file on a tape, and, on the other hand, a file may extend over a number of tapes.

A file is commonly understood to consist of a number of individual records, the records being generally similar to each other in size, content, and format. (Sometimes a series of differing records may be grouped together in a file; in such a case, the user must make provisions for distinguishing among them.) The file itself may be unique, or it may closely resemble other files. Thus, a file of actuarial data might be the only one of its kind in a library, while a file of insurance policy data might differ from another chiefly because it covers a different area or a different period of time. In this sense, a file serves a function like that of a ledger, or a filing cabinet containing a series of similar papers. It is usually a rather large body of related information. Thus it is convenient in most cases to treat such large bodies of data as complete and separate units which can be identified externally by direct reference to the physical media in which they are stored.

ELEMENTS AND GROUPS OF ELEMENTS An element of data (elementary item) is a piece of data which is never further divided. DATE could be the name of an elementary item if it were never referred to as anything but DATE. However, if the Decision Tables ever referred to only a part of the date, say, the month, then DATE would have to be broken into parts. These parts might be named DAY, MONTH and YEAR. Then DATE would not be the name of an element, because it is subdivided into DAY, MONTH and YEAR.

The term ITEM, as used in this manual, refers to any element or group of elements. Thus, it is correct to say that DATE is the name of an item. However, it is not the name of an elementary item, because the item is actually a group - DAY, MONTH, and YEAR. The terms ITEM and FIELD are used interchangeably.

An item can also be a group of groups. Suppose the item YEAR in the previous example were divided into two parts, named DECADE and YR, for example. YEAR would not be the name of an element - it would be the name of an item consisting of a group of elements. Then DATE would be the name of an item that was a group containing a group.

To review, elements can be combined to form groups which, in turn, can form groups of groups. Either an element or a group may be regarded as an item. An element is a unit of data which is never broken into smaller units. The discussion of levels (later in this chapter) will help to clarify the concept of elements and groups of elements.

Elements and groups of elements of data are combined to make up records. In a program processing a payroll, the permanent data concerning each employee would probably constitute a single record. Items in this record might be the employee's man number, his name, shift, rate of pay, marital status, number of dependents, etc. Thus, there would be a series of similar records, one for each employee. All of these records would usually be of the same format, i.e., would contain the same items; but the items would have different values for each employee.

Probably, they would all be stored together on a single reel of magnetic tape or in a single deck of cards. Another series of records (again, one for each employee) might contain the record of each man's time card for the past week. Thus, the payroll program would have two records available pertaining to each employee, one containing permanent information, the other containing the record of the employee's past week of work.

An important characteristic of the record is that it is the unit of data which is handled by the READ and WRITE verbs. (See Chapter 4.) If, in the above example, the records containing the permanent information for all employees are stored in a file called PERMANENT.PAYROLL.INFORMATION, then each time the statement READ PERMANENT.PAYROLL.INFORMATION RECORD is encountered, one man's permanent information would become available for processing. That is, it would be "read" into the system. It is not possible to read only a fraction of a record and not the rest of the record. Similarly, only an entire record can be "written," that is, made available for output to an external medium such as magnetic tape.

DATA DESCRIPTION FORMAT. Entries in each field shall be described completely except for explaining most of the report description. It will be shown at the end separately.

The heading information at the top of the form is completed similar to the corresponding fields on the decision table explained in Chapter 4.

PAGE Enter number of the data description.

TO PAGE. Enter page number of the next page of the data description. The use of this field allows insertion of new pages (numbered with a decimal system) with a minimum effort and still allows a user to know all pages are present for the data description. For the last page enter the word FINAL.

CTL. and SERIAL. It is essential that each item of the data description be entered into the system in proper sequence, since the sequence controls the internal position of the data. Six characters are used for a serial number, which indicates the sequence of the lines. This number is normally numeric. Its first three digits are written in the box marked "CTL" (for "control"). It is assumed that the first three digits will be common to all serial numbers written on the same page.

The remaining digits are written in the box labeled "SERIAL". In normal practice, only the left two digits are used initially, and the right most digit left blank. This makes it possible to insert correction lines later.

DATA NAME is any name the user may have assigned to the data described in the line. The rules for forming names are on page 3.3. If no name is assigned the field should be left blank.

DECISION TABLE DATA DESCRIPTION

CTL	SYSTEM			SYSTEM SEGMENT	PREPARED BY	DATE	PAGE TO PAGE
SERIAL	NAME	LEV-EL	TYPE	DESCRIPTION			CONT

FIGURE 5.1
Sample Form of Data Description

Names may be assigned to any item of data, or to any group of data items stored consecutively within the system. Thus, names may be given not only to groups of items in the input files, but also to groups formed within storage as a result of operations performed by the system. Any name so assigned may be used as an operand in a decision table.

Data-names must not overlap. Each field within a record can be given a name, and any group of consecutive fields can also be given a name. Thus, a single field may be operated on individually by reference to its name, or collectively as part of a group called by the group name. However, the same field may not be included as a part of each of two overlapping named groups of fields.

For example, if three successive fields are named A, B, and C, the group name X might be assigned to the pair A and B. If this were done, the name Y could not be assigned to the pair B and C, since field B is already part of a named group of fields. If the user needs to be able to refer to fields B and C by single name, however, he can rename the entire group of three fields, using the REDEF type code described later in this chapter. This procedure would not delte the original names; the new names and the names originally assigned would all be available for use thereafter.

LEVEL Level numbers are used to describe the way in which a body of data is organized. Basically, level numbers are assigned to items of data to show their relationship to other items of data--or, in other words, to show the structure of a record. Any number from 1 to 99 can be used. All data description entries must be assigned level numbers.

In general, each item is considered to be a subdivision of the last item preceding it which has a lower number. Figure 5.2 shows how a typical series of files, records, and fields might be organized, using the familiar method called outlining. The file structure is shown by the use of indentation, each item being considered a part of the last item above it which is indented to a lesser degree.

The technique of indentation, in other words, is a visual way of showing level. It may be used in the "Data Name" columns, but it will have no effect on the implementation. However, since it helps to identify the various levels visually, indentation may be useful in clarifying the file structure.

For comparison, two additional columns have been provided at the right in Figure 5.2. These show the data classification of each item, together with hypothetical level numbers such as might be assigned to a file structure of this kind. It should be pointed out that entries for files and groups of files are not actually used in the data description.

It is obvious from the outline that each item from EMPLOYEE NUMBER through LOCATION is a part of PAY RECORD, and that each item from FICA through HOSPITALIZATION is a part of DEDUCTIONS, YEAR TO DATE. Had the principle of indentation not been used, the reader might still determine these relationships by examining the level numbers in the right hand column, following the rule that each item is part of the next item above with a lower number.

It is not necessary that level numbers be assigned in consecutive order, although it is done that way in Figure 5.2. The items at level 02, for example, might have been assigned level 04, or any other convenient number, as long as it was greater than 01. Similarly, items at level 03 could have been given any other number as long as it was greater than the number of the next higher classification. In fact, it is often useful to skip numbers when they are initially assigned, to allow for possible regroupings or insertions at a later time.

The reader will also note that each item at the record level and below represents a kind of data, not a specific item of information. Thus, although there will be only one file called EASTERN REGION SALES FORCE, within that file there will be many individual units called PAY RECORD, and each of these will contain information of the same general character and format, as specified by the names of fields within it. The purpose of the data description is to give information about each of these kinds of data. The data description should be thought of as a "pattern" which the files will follow.

Level numbers are not actually attached to the data in the sense that an employee number is part of a pay record. They are used to instruct the implementor to perform certain technical functions which need not concern the user. Essentially, they are used before the actual data is read into the system, as a means of preparing the system to receive it. Once the data description has been written, the user need no longer concern himself with level numbers unless, owing to changes in the data or the problem definition, a new data description should become necessary.

TYPE This field is used, when necessary, to show that the data being described is of a certain special type. If left blank, it will be assumed that the remainder of the particular entry describes a data field or group of fields. The type codes which may be used in these columns are the following:

RECORD
COND
REDEF
COPY
WORK
EXPRES

Each of these is discussed in the following pages.

RECORD This type code shows that the data being described is a record and is therefore accessible by READ and WRITE COMMANDS. This is equivalent to identifying an item of data as an input/output record.

COND The type code COND is used to show that the data referred to is one of the possible conditions which a conditional variable may assume. In the discussion of conditional expressions in Chapter 3, it was pointed out that a conditional variable is the name of a field which will contain, at different times, any of a number of different values, depending on conditions existing in the data. Each of the values that may be placed in the field is a "condition."

ORGANIZATION OF PAYROLL FILES

Standard Outline	Equivalent in this language: Data Classification	Level Number
*EASTERN REGION	group of files	
*SALES FORCE	file	
PAY RECORD	record	01
EMPLOYEE NUMBER	field	02
EMPLOYEE NAME	field	02
LAST NAME	field	03
FIRST NAME	field	03
JOB TITLE	field	02
COMMISSION RATE	field	02
GROSS PAY, YEAR TO DATE	field	02
DEDUCTIONS, YEAR TO DATE	field	02
FICA	field	03
FEDERAL INCOME TAX	field	03
STATE INCOME TAX	field	03
SAVINGS BONDS	field	03
HOSPITALIZATION	field	03
NET PAY, YEAR TO DATE	field	02
LOCATION	field	02
*PRODUCTION FORCE	file	
PAY RECORD	record	01
EMPLOYEE NUMBER	field	02
EMPLOYEE NAME	field	02
LAST NAME	field	03
FIRST NAME	field	03
JOB TITLE	field	02
HOURLY RATE	field	02
GROSS PAY, YEAR TO DATE	field	02
DEDUCTIONS, YEAR TO DATE	field	02
FICA	field	03
FEDERAL INCOME TAX	field	03
STATE INCOME TAX	field	03
SAVINGS BONDS	field	03
HOSPITALIZATION	field	03
NET PAY, YEAR TO DATE	field	02
LOCATION	field	02
*WESTERN REGION	group of files	
*SALES FORCE	file	
PAY RECORD	record	01
EMPLOYEE NUMBER	field	02
⋮		
LOCATION	field	02

*Files and groups of files are not actually entered as such in the data description. Also, none of the names is in the proper format.

FIGURE 5.2. Typical File Structure (theoretical)

In the following example, the name MARITAL.STATUS is given as the name of a conditional variable. This name refers to a specific field reserved in storage into which values representing conditions will be entered. Typical conditions for this field would be "single," "married," and "divorced." While these words could actually be placed in the MARITAL.STATUS field, it is more economical of space, and generally more efficient, to use codes. The inititial letters M, S, and D are used as codes in this example. Thus, the field MARITAL.STATUS might contain any one of these letters at a given time.

However, so that the user can refer to these codes by their names, he must specify in the data description which code corresponds to each name. This may be done in the following manner:

Suppose that the field MARITAL.STATUS has been given the level number 06. The names of the conditions which may be entered into the field must then be assigned a lower level (i.e. a higher number) and entered in the data description immediately following the name of the field. This means that they will be treated as if they were each a subdivision of the field, in accordance with the rules for assigning level numbers, although, in practice, only one condition will be considered at a time. A portion of the data description might then appear as follows:

Serial	Data Name	Level	Type	Description
	MARITAL.STATUS	06		A
	SINGLE	07	COND	'S'
	MARRIED	07	COND	'M'
	DIVORCED	07	COND	'D'

The entries under "Description" will be explained later in this chapter, but, in summary, the "A" indicates that the field will contain one letter of the alphabet or a blank, while the initials S, M, and D are enclosed in quotation marks to show that they are the actual values to be used in the program.

In this example, the fact that the names SINGLE, MARRIED, and DIVORCED are the names of conditions is shown by the use of the type code COND. The relationship of these conditions to the field MARITAL.STATUS is shown by the fact that the condition-names have a higher level number and follow the name of the conditional variable immediately.

It should always be remembered that the condition-name is the name of the "value" which can be placed in a field; it is not the name of the field itself. The condition-name MARRIED, in this case, would be equivalent to MARITAL.STATUS='M'.

REDEF The code REDEF is used whenever it is necessary to redefine an area or an item of data that has previously been defined in some other way. This is usually necessary whenever a portion of the data description "overlaps" another--i.e., when it calls for the use, on a "time-sharing" basis, of data or storage space which has previously been defined for some other purpose.

For example, it may be necessary to call existing data by a new set of names, or to reorganize it by altering the groupings and/or the subordinate level numbers. Frequently it is necessary to wipe out data to make room for other data. In any such case, a new data description is required for the new items or the new names. However, the name or names of the areas being redefined must first be listed, using the type code REDEF to show that the accompanying data description may also be used to refer to the same area. The REDEF entry must have the same level number as the entry being redefined.

Use of the REDEF code does not erase data in storage, unless an attempt is made to place two or more different constants in the same area; however, it does superimpose a new format upon the data already present. If the user wishes to change an item in storage, he may do so by using a MOVE or SET instruction that specifies the new data and the position in storage where it is to be placed.

Redefinition does not cancel the previous definition. It merely makes it possible to refer to the same area by different names and for different uses. Once an area has been defined, all names associated with the definition may be used at any time, regardless of subsequent redefinitions.

Data Name	Level	Type	Description
RECPT	01	RECORD	
TR.CODE	02		
STOCK	02		
LOC	02		
QTY	02		
	02		
ISSUE	01	REDEF	RECPT
TR.CODE	02		
STOCK	02		
LOC	02		
QTY	02		
DATE	02		

In the illustration above the two types of records are read from a transaction file into the same area of storage. The REDEF is used in the ISSUE description to overlap the storage area reserved for RECPT.

COPY This type code is used to copy a data description previously defined in the data description so that it can be used again elsewhere. This makes it possible to use a data description with new data-names and, if desired, new level numbers.

The COPY type code is used as follows: The new name of the data description entry is written in the "Data Name" column of the new entry. The code COPY is placed in the "Type" column. The data description to be copied is specified by writing its original name in the "Description" column. This description must already have been read into the system for the COPY code to be able to operate on it.

The implementor will then obtain the original data description and copy it in its entirety, except for the following modifications: (1) The original name will be replaced by the new name. (2) If a new level number has been specified for the new name, the level numbers of the original data description will be adjusted so that they retain their original relationship to the named entry. Thus, if the original sequence of level numbers had been 01, 03, 04, and if the new name is assigned level 05, the other items would now be placed at levels 07 and 08, respectively.

Suppose the user had previously written the following entries in the data description:

Serial	Data Name	Level	Type
	PAY.RCD.MASTER	01	
	EMPLOYEE.NAME	02	
	JOB.TITLE	02	
	HOURLY.RATE	02	
	GROSS.PAY	02	
	TAXES	02	
	FICA	03	
	FED.INCOME	03	
	STATE.INCOME	03	
	NET.PAY	02	

Suppose then that he wishes to set up an identical data description for a detail record, except that the new description is to have the name PAY.RCD.DETAIL and it will be placed at level 02. He could write the following entry:

Serial	Data Name	Level	Type	Description
	PAY.RCD.DETAIL	02	COPY	PAY.RCD.MASTER

The effect of this entry would be as though the programmer had written an entirely new set of entries in the following form:

Serial	Data Name	Level	Type
	PAY.RCD.DETAIL	02	
	EMPLOYEE.NAME	03	
	JOB.TITLE	03	
	HOURLY.RATE	03	
	GROSS.PAY	03	
	TAXES	03	
	FICA	04	
	FED.INCOME	04	
	STATE.INCOME	04	
	NET.PAY	03	

WORK The type code WORK is used to describe areas of storage where intermediate results and other items are stored temporarily at object time. Initial values may be assigned to an item in the working storage with the following restrictions:

1. The value must be compatible with the CLASS of the item. For example, if the CLASS is NUMERIC, only a numeric value may be assigned as an initial value.

2. The size of the initial value must not exceed the SIZE of the item. If the size of the value is less, standard rules for justification apply.

3. If the initial value of a work area is not specified by a VALUE, its, initial value will be unpredictable at object time.

Any item within a work area may be assigned condition-names.

Data Name	Level	Type	Description
STOCK	01	WORK	
PREV.BAL	02		9(6) ZEROS
BEG.BAL	02		9(6) '000000'

EXPRES The type code EXPRES is used to describe an arithmetic expression. The expression is always one of equality. The expression name is entered as level 01. The expression is entered as level 02 and can extend over as many lines as necessary to accommodate the data. The expression OK=IN.PROC AMT + ACCT.REC AMT + ORD AMT would be entered as follows:

Data Name	Level	Type	Description	Cont
OK	01	EXPRES		
	02		IN.PROC AMT + ACCT.REC	C
	02		AMT + ORD AMT	

Of course, the names used in the expression must be defined in the data description elsewhere.

DESCRIPTION This field is used to show:
1. Picture-Format characters. These are explained below.
2. Initial values of work areas or constants.
3. Data names associated with type codes REDEF and COPY.

General Note: If the description of a data item overflows from the "Description" column, it may be continued on the next line, following the rules given for the continuation indication column. The break at the end of a line must occur between words. If a constant or initial value is to be carried over onto a new line, the portion on each line must be treated as a complete constant (i.e., enclosed in quotation marks); the continuation indication is not used in this case.

In a number of cases, a complete data description entry will require that more than one of these kinds of information be listed on the same line. For example, it is generally necessary to show both the format and the value of a constant. In such a case, the various items should be written in the order shown above, separated by one or more blanks.

The picture characters serve two basic functions: (1) They show the number of character spaces to be occupied by a field. (2) They show the kind of character that will occupy each space.

If the item of data being described is one which will be brought into the system at object time, the format characters must reflect the format of the data as it already exists; changes in input data cannot be effected by the picture. However, if the item is one produced as a result of the operation of the program--as in moving the data or performing arithmetic on it, for example--the picture has a direct effect on the manner in which the data will be handled.

With certain exceptions, which are explained below, one format character is required for each data character for which storage space is to be reserved. The particular format character chosen for each space prepares the system to receive in that space data of the type explained below.

The PICTURE clause can be used to specify the editing of data. Editing may be described as an alteration of the format and/or punctuation of an item, usually for such purposes as improving readability or protecting it against unauthorized alteration. Editing involves the suppression of certain characters and/or the addition of others. For example, after computation, the digits representing a man's pay might be 0012531. However, they would be much more readable on

a paycheck in an edited form, such as $**125.31; moreover, the use of the asterisks would hamper an attempt to alter the amount. Editing of data always requires moving it to an item for which the proper editing symbols have been specified.

In the following discussion, each character which may appear in a PICTURE is presented. Because the choice of characters in any given PICTURE depends on the type of data item being described, the characters will be grouped for discussion according to the type of data item they describe.

A NUMERIC ITEM is an item which may contain only the numerals 0 through 9 and an operational sign. As will be seen below, a numeric item may also have an assumed decimal point associated with it. The PICTURE of any numeric data item may contain combinations of only the following characters: 9, V, P, S, T, L and R. An explanation of each of these characters and their uses is given below.

Character	Meaning and Use
9	A 9 indicates that the character position will always contain a NUMERIC character.
V	A V indicates the position of an "assumed decimal point." Since a numeric item cannot contain any character other than numerals and an operational sign, the actual decimal point (the special character period) cannot appear. Therefore, an assumed decimal point is used to provide the implementor with information concerning the alignment of items involved in computation. An assumed decimal point, thus, does <u>not</u> occupy a character position at object time and is not counted in the size of an item. For example, if a data item is described as having a PICTURE of 99V9 and the digits 123 are moved to it, the value would be treated in calculation as 12.3, but the size of the item would be three characters, not four. If it were printed, it would print as 123 because the decimal point character is not actually present.
S	The character S indicates the presence of an "operational sign." If used, it must always be written as the leftmost character of the PICTURE. It is <u>not</u> counted in the size of an item.
P	The character P indicates an "assumed decimal scaling position." It is used to specify the location of an assumed decimal point when the point is not within the number as it appears in "input data." In effect, the item will be treated as if a zero were substituted for each P and the decimal point were placed "outside" the last P--i.e., to the right if the zeros are on the right, to the left if the zeros are on the left. The character V may be used or

Character	Meaning and Use
	omitted as desired. If it is used, it must be placed in the position of the assumed decimal point. For example, an item composed of the digits 123 would be treated by an arithmetic procedure statement as 123000 if the PICTURE were 999PPPV or as .000123 if the PICTURE were VPPP999. The character P is never considered as part of the size of an item; in the above examples, the size would be three characters.
T	Truncate. This symbol is optional and does not reserve an actual space in storage; it informs the implementor that when data is moved into this field with too many digits, the low order numbers are truncated rather than the usual rounding. It is placed after the above format characters and before the justify character.
L or R	Justify. These symbols are optional and do not reserve an actual space in storage. L means left justify. R means right justify. This entry need be used only when the user wants the justify different from the normal assumed justify. An item of data may be moved within the system by means of a MOVE or SET or as a result of computation or some other operation. If the location to which it is moved is larger in size than the data itself, it may be necessary to specify the position the data is to occupy in its new location. In the absence of instructions to the contrary, NUMERIC data will be "right justified" under these circumstances, unless an assumed decimal point alignment occurs. When an item is right justified, its rightmost character will be placed in the rightmost position of the new location, and any unused positions at the left will be filled with zeros. ALPHABETIC and ALPHANUMERIC data, on the other hand, will be "left justified" in the absence of instructions to the contrary and any unused character positions at the right will be filled with blanks.

An ALPHABETIC item can contain only the letters of the alphabet and the blank. The PICTURE of an ALPHABETIC item can contain only the character A.

Character	Meaning and Use
A	The character A, when used in a PICTURE, indicates that the character position will always contain either a letter or a blank.

An ALPHANUMERIC (ALPHAMERIC) item is an item which may contain any character in the character set. However, it is often convenient to think of ALPHANUMERIC items as being divided into two types: "non-report" items and "report" items. Non-report items are items for which editing is not specified. Report items are items for which editing has been specified.

The PICTURE of an ALPHANUMERIC "non-report" item may contain only the characters L, R, 9, A, and X. The character L, R, 9, and A have been discussed above.

Character	Meaning and Use
X	The character X, when appearing in a PICTURE, indicates that the character position may contain any character in the character set. For example, the PICTURE AAAXXXX indicates that the described item has a size of seven characters, that the first three characters will always be alphabetic, and that the last four characters may be any characters.

It may be desirable to edit data which is being prepared for printing. Editing involves the insertion of certain characters and/or the suppression of others. Editing of data is accomplished by moving the data to a "report" item. A report item is an ALPHANUMERIC item governed by the following rules:

1. Any data which is moved to a report item is automatically altered according to the editing specifications given in the Data Description entry corresponding to the item. Editing is specified by means of a PICTURE.

2. A report item can receive only data which is numeric in content.

The characters which may appear in the PICTURE of a report item are shown below.

L R T P 9 V , . + - Z * CR DB B 0 $

All the characters in the PICTURE of a report item, with the exception of L, R, T, P, and V, must be counted in determining the size of the item. The uses of L, R, T, P, 9, and V have been discussed above. The remainder of the characters will be explained in three groups, zero suppression, insertion, and replacement characters.

Zero suppression and replacement characters are used to suppress and/or replace characters in accordance with the rules given in this section. Two general rules apply to these characters, as follows: (1) Except in the cases of 0 and B, suppression and/or replacement terminates with the character immediately preceding the first digit other than 0, or the decimal point, whichever is encountered first; e.g., zeros following a significant digit will not be suppressed or replaced. (2) If all data character positions in a PICTURE reserved for source data (as opposed to those additional positions used for insertion characters) contain suppression and/or replacement characters (other than 0), then all characters will

be replaced by blanks if the value of the data is zero. Note that this rule is equivalent in effect to the BLANK clause.

ZERO SUPPRESSION CHARACTER

Character	Meaning and Use
Z	The character Z specifies "zero suppression" of the indicated characters. Zero suppression is the process of replacing unwanted left-hand zeros by blanks. The following table indicates the effect of zero suppression:

Source Item	Editing PICTURE	Edited Item
12345	ZZZ99	12345
00123	ZZZ99	123
00100	ZZZ99	100
00000	ZZZ99	00
00100	ZZZZZ	100
00000	ZZZZZ	100

A Z must never be preceded by a 9, a B, or a 0.

An "insertion" character is one which is inserted into a report item. An insertion character does not take the place of any data; it appears in addition to the information moved to the item. The insertion characters are $, . . + - CR DB. When any of these characters is used, the size of the report item must be larger than the maximum number of digits which might be moved to the item. This principle is illustrated in the discussion of the dollar sign below.

Character	Meaning and Use
$	The single dollar sign, placed in the leftmost position of a PICTURE, specifies that a dollar sign character is to be placed in that position in the data, as illustrated in the following table:

Source Item	Editing PICTURE	Edited Item
123	$999	$123
012	$999	$012
012	$ZZZ	$ 12
000	$ZZZ	
010	$ZZZ	$ 10

Note that the PICTURE of the item specifies four character positions; however, a maximum of three digits of data can be moved to the item.

Character	Meaning and Use

-

If the minus sign is written as either the first character or the last character of a PICTURE, a "display" minus sign (as opposed to an operational sign) will be inserted into the indicated character position when the value of the item is negative. If the value of the item is not negative, a blank will be inserted. Consider the following:

Source Item*	Editing PICTURE	Edited Item
1$\bar{2}$	-99	-12
12	-99	12
1$\bar{2}$	99-	12-
12	99-	12
0$\bar{0}$	99-	00-
00	99-	00

+

If the plus sign is written either as the first character of a PICTURE or as the last, a "display" sign will be placed in the indicated character position. If the value of the item is negative, a minus sign will appear; otherwise, a plus sign will be inserted. If an item is unsigned, it is assumed to be positive. The following table illustrates the above principle:

Source Item*	Editing PICTURE	Edited Item
1$\bar{2}$	+99	-12
+ 12	+99	+12
1$\bar{2}$	99+	12-
+ 12	99+	12+
0$\bar{0}$	99+	00-
+ 00	99+	00+

*A sign over the units position of a number indicates an operational sign.

,

The comma, when used to describe a character position, will be inserted at the indicated position in the data being edited. For example, the PICTURE 9,999 would cause 7461 to become 7,461 after editing. The comma itself will be suppressed if zero suppression has caused the elimination of all digits to the left.

Character	Meaning and Use
.	This character represents an "actual decimal point." When used to describe a character position: 1. The data being edited is aligned by decimal point. 2. An actual decimal point will appear in the indicated character position. Thus, the integer 7531 would appear as $7,531.00 if the notation $9,999.99 were used as the editing PICTURE. Unlike the assumed decimal point, the actual decimal point occupies a character position and is counted in determining the size of an item. A PICTURE may never contain more than one decimal point, assumed or actual.
CR and DB	The "credit" and "debit" symbols CR and DB may appear only at the right end of a PICTURE. These symbols occupy two character positions. When the value of the described item is negative, the specified symbol will be placed at the right end of the item. If the value of the item is positive, these characters will contain blanks. For example, the PICTURE $99.99CR will cause 6325 to become $63.25CR and 6325 to become $63.25 after editing. The only characters which can appear to the right of CR and DB are T, L, and R.

Several of the characters used in PICTURE specify that, at object time, certain digits will be replaced by other characters much in the same way that the Z specifies the replacement of leftmost zeros with blanks. The list of "replacement" character consists of: *, 0, B, the floating dollar sign, the floating minus sign, and the floating plus sign.

Character	Meaning and Use
*	The asterisk is used to indicate "check protection," i.e. the suppression of each specified zero on the left and its replacement by an asterisk. The following table illustrates the use of the asterisk.

Source Item	Editing PICTURE	Edited Item
12345	***99	12345
00123	***99	**123
00100	***99	**100
00000	***99	***00
00000	*****	
00100	*****	**100

Character	Meaning and Use
	An asterisk can be preceded only by a dollar sign, a plus sign, a minus sign, a decimal point, or a comma.
0 (zero)	The character 0 (zero) will cause a zero to replace whatever character formerly occupied the indicated character position. For example, if the digits 123456 are to be moved to an item with a PICTURE of 999000, the item will appear as 123000.
B	The character B specifies that a blank will replace whatever character formerly occupied the indicated character position. For example, if the digits 123456 are to be moved to an item with a PICTURE of BB9999, the item will appear as 3456.
The floating dollar sign	Zero suppression with a "floating dollar sign" is specified by placing a dollar sign in each leading numeric character position to be suppressed. A dollar sign will be placed in the rightmost position in which suppression by a dollar sign is to occur. The following table illustrates the principle:

Source Item	Editing PICTURE	Edited Item
123	$$99	$123
012	$$99	$12
001	$$ZZ	$ 1
000	$$$$	

The floating minus sign is similar in explanation to the floating dollar and plus sign.

The floating plus sign	Zero suppression by means of a "floating plus sign" is specified by placing a plus sign in each leading numeric character position to be suppressed. If the value of the item is negative, a minus sign will be placed in the rightmost position in which suppression is to occur; if the value of the item is not negative, a plus will be inserted instead. The following examples illustrate the effect of the floating plus sign:

Source Item	Editing PICTURE	Edited Item
123	++99	+123
012	++99	+12
012	++99	-12
001	++99	-01
000	++++	

All floating plus signs must be the leftmost characters in a PICTURE.

General notes on the PICTURE.

1. When an integer is placed in parentheses immediately following a PICTURE character, it indicates the number of successive times that character is to be present. For example, the notation P(4) 9(10) is equivalent to PPPP99999 99999 and will be interpreted in the same way. The parentheses must follow the indicated symbol without an intervening space.

2. The number of characters in a PICTURE must not exceed 30. For example, $$ZZ.99 is a PICTURE containing seven characters, 9V9 contains three characters and 9(1000) contains seven characters. Thus, the number of characters in a PICTURE may be different than the number of character positions described by the PICTURE.

The following are examples of applications of PICTURE which do not contain editing symbols. A sign over the units position of a number indicates an operational sign as opposed to a display sign.

Non-Editing Applications

If PICTURE is:	and the characters in the item are:	Then the item will be used in procedures as:	and its class will be:
99999	12345	12345	NUMERIC
999V99	12345	123.45	NUMERIC
S999V99	1234$\bar{5}$	123.4$\bar{5}$	NUMERIC
S9(3)V9(2)	1234$\bar{5}$	123.4$\bar{5}$	NUMERIC
XXXXX	12345	12345	ALPHANUMERIC
AAAAA	ABCDE	ABCDE	ALPHABETIC
XXXXX	ABCDE	ABCDE	ALPHANUMERIC
999X99	123.45	123.45	ALPHANUMERIC
999AA	123AB	123AB	ALPHANUMERIC
999XX	123AB	123AB	ALPHANUMERIC
XXXAA	123AB	123AB	ALPHANUMERIC
XXXXX	123AB	123AB	ALPHANUMERIC
9(3)A(2)	123AB	123AB	ALPHANUMERIC
99PPP	12	12000	NUMERIC
99PPPV	12	12000	NUMERIC
P(3)9(2)	12	.00012	NUMERIC
VP(3)9(2)	12	.00012	NUMERIC

The examples which follow illustrate the use of PICTURE to edit data. In each example a movement of data is implied, as indicated by the column headings.

Editing Applications

Source Area		Receiving Area	
PICTURE	DATA	PICTURE	EDITED DATA
99999	12345	$ZZ,ZZ9.99	$12,345.00
99999V	00123	$ZZ,ZZ9.99	$ 123.00
9(5)	00100	$ZZ,ZZ9.99	$ 100.00
9(5)V	00000	$ZZ,ZZ9.99	$ 0.00
9(5)	00000	$ZZ,ZZZ.99	.00
9(5)	00000	$ZZ,ZZZ.ZZ	
999V99	12345	$ZZ,ZZ9.99	$ 123.45
V99999	12345	$ZZ,ZZ9.99	0.12
9(5)	12345	$**,**9.99	$12,345.00
9(5)	00123	$**,**9.99	$***123.00
9(5)	00000	$**,***.99	$*****.00
9(5)	00000	$**,***.**	
9(5)	00000	$**,***.ZZ	
99V999	12345	$**,**9.99	$****1.35
V99999	12345	$**,**9.99	$***0.12
9(5)	12345	$$$,$$9.99	$12,345.00
9(5)	00123	$$$,$$9.99	$123.00
9(5)	00000	$$$,$$9.99	$0.00
9(5)	00000	$$$,$$$.ZZ	
9999V9	12345	$$$,$$9.99	$1,234.50
V9(5)	12345	$$$,$$9.99	$0.12
S99999V	12345-	-ZZZZ9.99	-12,345.00
S9(5)V	12345+	-ZZZZ9.99	12,345.00
S9(5)	00123-	-ZZZZ9.99	-123.00
S99999	12345+	ZZZZ9.99-	12,345.00
S9(5)	12345-	ZZZZ9.99-	12,345.00-
S9(5)	00123+	------.99	123.00
S9(5)	00001-	------.99	-1.00
S9(5)	12345+	+ZZZZZ.99	+12,345.00
S9(5)	12345-	+ZZZZZ.99	-12,345.00
S9(5)	12345+	ZZZZZ.99+	12,345.00+
S9(5)	12345-	ZZZZZ.99+	12,345.00-
S9(5)	00123+	++++++.99	+123.00
S9(5)	00001+	++++++.99	+1.00
9(5)	00123	++++++.99	+123.00
9(5)	00123	------.99	123.00
9(5)	00000	++++++.ZZ	
9(5)	00000	------.ZZ	
9(5)	12345	BB999.00	345.00
9(5)	12345	00099.00	345.00
S9(5)	12345-	$$$$$.99CR	$12,345.00CR
S9(5)	12345+	$$$$$.99CR	$12,345.00
9(3)V9(3)	123456	999.99T	123.45
9(6)V	123666	999PPP	123
9(6)V	123666	999PPPT	123
9(2)	88	99999L	88000

CONSTANTS and INITIAL VALUES in a WORK AREA A constant or initial value may be placed in the system by writing a data description entry for it which includes both a statement of its format (using the PICTURE characters) and a statement of the actual value or group of symbols. The format is specified by a standard PICTURE entry. The actual value, or the actual symbols, must then be written on the same line, separated from the PICTURE by at least one blank. This value (or this group of symbols) must be enclosed in quotation marks.

Data Name	Level	Type	Description
PERCENT			V99 '05'

In this example, the notation V99 is, of course, the PICTURE, and it is followed by a statement of the actual value of the constant.

Data Names Associated with REDEF and COPY When the type codes REDEF and COPY are used, it is necessary to specify the name of the data item or area to be redefined or copied. This name must be entered in the "Description" columns. (See the discussion of those type codes earlier in this chapter.)

REPORT GENERATION FEATURES (RGF) are included in this language to accomplish such functions as:

- Print Headings per page and per change in control fields data.
- Automatically increment and print page number.
- Print date on each page of report.
- Group suppress printing of control field data.
- Print selected field totals automatically per change in control field data and at end of report.
- Generally, move and edit fields into lines of print from fields of detail lines in working storage. Editing consists of functions such as suppress leading zeros, insert periods and commas in quantity fields, floating dollar signs and asterisk check protection.

These functions are accomplished, primarily, by data descriptions. The main flow of the problem definition prepares a working storage of unedited data for a detail line. The command DO REPT.n (where n is the report number) assumes that the detail line is edited for print in the manner specified by the data description including the consideration of all the other report functions defined in the data description. At the beginning of the report, the command DO BEG. REPT.n may be given to effect the initializing of the report such as writing a cover page, page heading, etc. At the end of the report the command DO END. REPT.n is given to effect the end of report functions such as printing all required levels of totals (including report total).

The RGF is explained in two parts with instructions for:

1. Completing the Printer Spacing Chart
   - Layout of lines and fields
   - Line identification code
   - Classification of lines

2. Completing the Data Description
   . Report Identification
   . Source of report data
   . Control fields in the source data
   . Header lines
   . Detail lines
   . Total lines

A sample report and its corresponding data description are shown (page 5.30 through 5.35) to illustrate the RGF instructions.

PRINTER SPACING CHART The purposes of laying out the report on the Printer Spacing Chart are:

1. to establish the positions at which the various data will be printed as well as to indicate the spacing between printed lines, and
2. to assign each line a unique identification code representing
   a. the type of line,
   b. the level of the line, and
   c. the number of the line within its level.

Layout of Lines and Fields The numbers across the top and bottom of the spacing chart represent the print positions. The numbers down the left side are line numbers. The user selects the line number and print positions for a particular field and makes his notation in the selected positions. In the sample layout (Figure 5.4) note that headings and other constant information are spelled out completely in the print positions assigned to them. Variable information is represented by X's and includes, where applicable, credit symbols, punctuation, etc.

Line-Identification Code The column at the left on the spacing chart is used to assign each line a three-character identification code. This code identifies the line on the data description sheet where each line is described according to type and content.

Type

The first character of the identification code is H for a heading line, D for a detail line, or T for a total line. All lines must be identified as belonging to one of these categories.

Level

The second character of the identification code can be a number or a letter. A heading or total line within a hierarchy is assigned a number to represent its level. A heading or total line that is independent (not in a hierarchy) is assigned a letter. A detail line can be assigned either a number or a letter to represent its level.

Number

Lines within a level can be numbered according to their appearance in the output. Lines with "numerical-level" designations must also have numerical line-number designations. Lines with "alphabetic-level designations can have numerical or alphabetic line-number designations.

Classification of Lines The classification of a line by type is usually self-explanatory. Note, however, that total lines differ significantly from heading and detail lines. Heading and detail lines can contain information from the record in the source area at the time when the lines are produced; total lines can not. A source record can effect the control-field change that causes total lines to be written, but the source record cannot contribute data to these lines. A detail line has a direct relationship to the source record in that most or all of the data in a detail line comes from the input record. A heading line generally contains constant information, although it can have some information from input records, including the record present at the time the line is assembled.

The concept of line "level" is based upon the relationship of a line to other lines. Heading or total lines that are independent of each other should be given alphabetic-level designations. Heading or total lines that are related in a hierarchy should be given numerical-level designations corresponding to their positions in the hierarchy. A hierarchical relationship can be likened to total operation on an accounting machine, i.e., major lines force minor and intermediate lines. The principle underlying a hierarchical relationship is that lines of higher level govern lines of lower level. Thus, when the object program is running, a total line with numerical-level designation such as T3x will force T1x and T2x to come before it whenever the output conditions are fulfilled for T3x.

In the "Monthly Expense Distribution Report" (Figure 5.3) there are three related levels of total. The lowest level is associated with sub-ledger number, the next level with general ledger number, and the highest level with department number. When the general ledger number changes, the sub-ledger total line prints before the general-ledger total line. When the department number changes, the sub-ledger and general-ledger total lines print before the department total line. Because this hierarchical relationship exists, these lines have been given the numerical-level designations T11, T21, and T31. (See the spacing chart in Figure 5.4.)

Study of the Monthly Expense Distribution Report reveals a difference in the hierarchical relationships for total and heading lines. Total lines appear in "ascending" order by level; heading lines appear in"descending" order by level. On that report the heading lines associated with department number print before the general-ledger number lines which, in turn, precede the sub-ledger lines. Thus, when department number changes, the H3x lines precede the H2x and H1x lines. When general ledger number changes, the H2x lines print before the H1x lines.

In addition to the three related levels of heading and total lines in the Monthly Expense Distribution Report, there are independent heading and total lines. Page headings printed as a result of page overflow, page totals, and final totals are all examples of lines that are independent of the minor, intermediate, and major relations governing a hierarchy. The illustrated report has both page heading lines (HB1-HB4) and a final total line (TAA). All of these have alphabetic level designations because they do not relate to lines of other levels.

Page heading lines may be printed because of control-field changes, because of page overflow, or because of both conditions. The latter case is true in the Monthly Expense Distribution Report. Just as the IBM 407 Accounting Machine distinguishes between normal programming and overflow programming, the RGF considers normal heading lines distinct from overflow heading lines, even when the lines are alike in format. Thus, some of the heading lines in Figure 5.4 have two names reflecting their status as both overflow and normal page-heading-lines. The normal heading lines are H3x, H2x, and H1x. The overflow lines are HBx. On a change in department number there is a skip to a new page of the report and the numerical-level heading lines (H3x-H1x) print. On page overflow the alphabetic-level heading lines HBx print.

When there is a single detail-line format in a report, that line can be given an alphabetic-level designation to reflect its independent status. Such is the case in the Monthly Expense Distribution Report in which the detail line is named DAA. Other applications might have any number of detail-line formats which, when they do not relate to one another, are classified alphabetically by level. It is conceivable that some applications might contain hierarchies of detail lines necessitating numerical-level designations.

Note that the "level" of a line is not necessarily equal to the "number of the control field" with which the line is associated. For instance, a total or heading line of level three may not relate to control field three in the input data file.

The "line number" permits scheduling lines within a level. The Monthly Expense Distribution Report has six heading lines composing level three print in line-number sequence within that level, i.e., H31, H32, H33, H34, H35, and H36. Even though there is only one line in each of the lower levels of heading line in that report (H21 and H11), the lines have numerical line-number designations because they are hierarchical. The same principle applies to the total lines, T11 T21, and T31, in the same report. Application of the line-number concept to hierarchical total lines corresponds to special programming on the IBM 407 Accounting Machine. For instance, four minor total lines could be named T11, T12, T13, and T14.

Analysis of the independent lines in the Monthly Expense Distribution Report reveals that both numerical and alphabetic line-number designations can be used in classifying lines with alphabetic "level" designations. When page overflow occurs, four heading lines (HB1-HB4) print, and the line number reflects the place of each line in the sequence. If there is only one line in an alphabetic level, that line can have a letter as its line number (DAA and TAA in the example report).

Multiple-line print (MLP) source date might cause three detail lines to print, and these lines could be named DA1, DA2, and DA3 to reflect the place of each line in the sequence. Two final total lines in a report might well be named TC1 and TC2.

DATA DESCRIPTION form is used for describing the source record in a work area and the lines and fields constituting the output.

FIGURE 5.5 shows an example of a report source area data description. FIGURE 5.6 shows an example of a report definition data description. Both figures correspond to figures 5.3 and 5.4 for the Monthly Expense Distribution. FIGURE 5.5 is completed in accordance with previous instructions in this chapter.

The report definition must be completed in the specific sequence shown below:

- Report identification
- Source
- Control fields
- Report lines

REPORT IDENTIFICATION Entries are:

DATA NAME	Condensed report name
LEVEL	01
TYPE	REPT.n when n is a unique number of this report.

SOURCE Entries are:

DATA NAME	The name of the source work area which will contain data for the report.
LEVEL	02
TYPE	SOURCE

CONTROL FIELDS If the data sequence in the source area is not a condition for printing lines, the control fields do not have to be specified. If they are specified they must be named in ascending order beginning with the minor control field as F1 (see Figure 5.6). Entries are:

DATA NAME	Control field name in source area.
LEVEL	03
TYPE	Fn where n is the control field level, starting with 1 as the minor level

REPORT LINES Entries for hierarchical line specifications must be in the same order as they will appear on the report. Entries for all of the fields within a line must follow the entry for that line specification. The field entries must be in the same sequence as they will appear within a line on the report. Entries for report lines are:

DATA NAME	No entry required.
LEVEL	02
TYPE	A three character entry (line identification code) is copied directly from the spacing chart as explained previously under Printer Spacing Chart.

The left character specifies the type of line (this is different from the type column on the data description form). It must be H for a heading line, D for a detail line, or T for total line. An important consideration in assigning "type" to a line is the difference between a total line and a heading or detail line with regard to the record in the source area when the line is formed. The performance of the total-line calculations, and the formation of total lines precede the function of removing fields from the source record. Thus, a source record that causes a control change cannot contribute data to total lines that result from that control change.

H and D lines follow the removal of fields from the record in the source area, and thus that record can contribute to these lines. Therefore, the naming of a line according to type is not arbitrary, particularly with regard to total and detail lines.

The line identification code must be in descending level-order for heading lines, and ascending level-order for total lines. This is the normal order of printing related report lines, as can be noted on the spacing chart in Figure 5.4 and on the printed report in Figure 5.3.

DESCRIPTION There are seven possible kinds of entries. They are listed below and must be entered in the sequence listed. It is very unlikely that all seven entries would be present at one time. A comma is written between entries.

1. NEXT LLL where LLL is the line identification code of the next line to be printed. This entry is made only if the next line specified by this entry should be printed unconditionally after this line. When a next line is specified it must be of the same type and level of the line being specified.

2. SPACE n BEFORE where the values for n are 01, 02, and 03 and specify single, double or triple line-spacing before printing the line being specified.

3. SPACE n AFTER where the values for n are 01, 02, and 03 and specify single, double or triple line-spacing after printing the line being specified.

4. SKIP n BEFORE where the values for n are 01 through 12 and specify skipping to carriage tape channels 1-12, respectively, before printing the line being specified.

5. SKIP n AFTER where the values for n are 01 through 12 and specify skipping to carriage tape channels 1-12, respectively, after printing the line being specified.

6. COND X This entry is used to designate the line output condition(s). If multiple conditions are designated on one line-entry, they are considered in an AND relation. The possible conditions (values for X) are:

OF	Page overflow
F1-Fn	Change in control fields 1 through n.
BEG.REPT.n	Begin report n. Used to initiate report functions such as writing a cover page and the first page headings.
END.REP.n	End report n. Used to accomplish end of report printing such as report total(s).
NOF NF1-NFn and NBEG.REPT.n NEND.REPT.n	These are negations of the above conditions.

If a line is referenced by a previous entry as a NEXT line, no conditions are entered.

A line will appear in the output only if:

1. "Line-output conditions" for the given line are specified and fulfilled, or
2. The line was specified as the next line of another line for which the output conditions are fulfilled, or

3. The line is of the lower level in a hierarchy than another line of the same type for which the output conditions are fulfilled.

A line conditioned by OF cannot be associated with any lines not conditioned by OF. Thus, a line conditioned by OF cannot specify a "next line" not conditioned by OF; nor can a line not conditioned by OF specify a next line conditioned by OF.

Lines related in a hierarchy must be of the same type and must have numerical-level designations that reflect their relative positions within the hierarchy. The processing principle underlying a hierarchical relationship is that lines of a higher level govern lines of lower level. Thus, when the output conditions are fulfilled for T3x lines, the object program will force T1x and T2x lines to come before the T3x lines in the output without regard for the line-output conditions of the T1x and T2x lines. This is precisely what happens in the Monthly Expense Distribution Report.

In addition, multiple lines within one level in a hierarchy must be referenced by "next line" designations rather than individual line-output conditions whenever there is a higher level of lines in the hierarchy. Thus, H21 must call for H22 as a next line and H22 must call for H23 as a next line to ensure that all three will be present in the output when H3x lines precede them.

7. COPY LLL This instruction accomplishes a similar function as the data description type COPY described previously. The LLL represents a previously defined line. The COPY LLL causes the field entries under line LLL to be copied exactly and placed as field entries under the current line being defined.

FIELDS WITHIN LINES Entries for line fields are:

DATA NAME	Entry required only when the field will contain variable data and thus must be referenced.
LEVEL	Entry will normally be 03. Lower level numbers (numerically larger) may be entered via the rules for the level numbers described earlier.
TYPE	No entry.
DESCRIPTION	The usual entries are made for fields - PICTURE which may be followed by a VALUE. An additional entry may follow the VALUE (or PICTURE if there is no VALUE). This entry is GROUP SUPPRESS which can be used only with control fields. It will cause group printing to be suppressed. Suppose a report appears as:

STOCK NR	LOCATION	CLASS	INVENTORY
111	AA	C1	100
111	AA	C2	150
111	AA	C3	50
111	BB	C1	20
111	BB	C2	30
111	BB	C3	40

If the fields STOCK NR and LOCATION were designated as GROUP SUPPRESS the report would appear as:

STOCK NR	LOCATION	CLASS	INVENTORY
111	AA	C1	100
		C2	150
		C3	50
	BB	C1	20
		C2	30
		C3	40

MONTHLY EXPENSE DISTRIBUTION REPORT

REPORT DATE 07-18-60 PAGE 1

	OUR INVOICE NUMBER	DATE MO	DAY	AMOUNT	AMOUNT BY ACCOUNT	AMOUNT BY DEPT
*** DEPT. NO. 042						
** GEN. LEDGER NO. 901						
* SUB. LEDGER NO. 615						
	12095	5	08	125.03		
				125.03 *		
* SUB. LEDGER NO. 623						
	12091	6	10	571.00		
	12088	5	16	685.94		
				1,256.94 *		
* SUB. LEDGER NO. 629						
	12080	5	03	24.15		
	12073	5	02	1,631.17		
				1,655.32 *		
* SUB. LEDGER NO. 636						
	12109	7	03	1,725.54		
				1,725.54 *		
					4,762.83	
** GEN. LEDGER NO. 906						
* SUB. LEDGER NO. 643						
	12150	6	08	402.00		

FIGURE 5.3
Monthly Expense Distribution Report

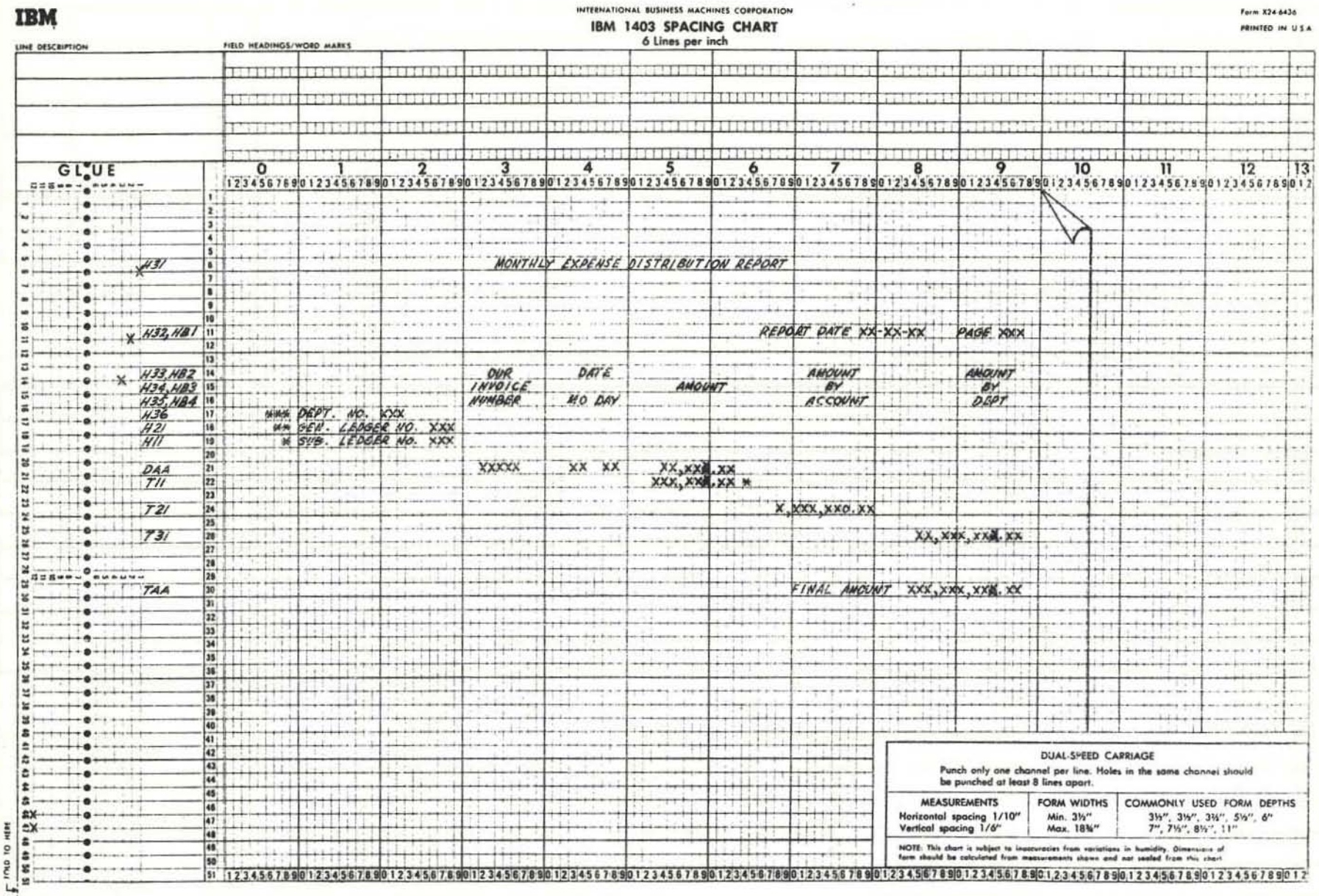


FIGURE 5.4
Spacing Chart for Monthly Expense Distribution Report

DECISION TABLE DATA DESCRIPTION

CTL 099	SYSTEM ACCOUNTING			SYSTEM SEGMENT MONTHLY EXPENSE DISTR	PREPARED BY OYE	DATE 8/1/61	PAGE 21 TO PAGE 21.1
SERIAL	NAME	LEVEL	TYPE	DESCRIPTION			SORT
01	DETAIL	01	WORK				
02	DEPT	02		9(3)			
03	GEN.LEDGER	02		9(3)			
04	SUB.LEDGER	02		9(3)			
05	INV.NR	02		9(5)			
06	DATE	02		x(6)			
07	INV.AMT	02		9(7)			

FIGURE 5.5
Report Source Area Data Description.

DECISION TABLE DATA DESCRIPTION

CTL 100	SYSTEM ACCOUNTING			SYSTEM SEGMENT MONTHLY EXPENSE DISTR	PREPARED BY OYE	DATE 8/1/61	PAGE 21.1 TO PAGE 22
SERIAL	NAME	LEVEL	TYPE	DESCRIPTION			CONT
01	MON.EXP.DIST	01	REPT.1				
02	DETAIL	02	SOURCE				
03	SUB.LEDGER	03	F1				
04	GEN.LEDGER	03	F2				
05	DEPT	03	F3				
06		02	H31	NEXT H32, SKIP 01 BEFORE, SIP 02 AFTER, COND F3			
07		03		A (33) BLANKS			
08		03		A (35) 'MONTHLY EXPENSE DISTRIBUTION REPORT'			
09		02	H32	NEXT H33, SKIP 03 AFTER			
10		03		A (65) BLANKS			
11		03		A (12) 'REPORT DATE '			
12	REPT.DATE	03		X (08)			
13		03		A (09) ' PAGE '			
14	PAGE.REPT.1	03		9 (03) '001'			
15		02	H33	NEXT H34, SPACE 01 AFTER			
16		03		A (32) BLANKS			
17		03		A (15) 'OUR DATE'			
18		03		A (24) BLANK			
19		03		A (06) 'AMOUNT'			
20		03		A (13) BLANKS			
21		03		A (06) 'AMOUNT'			
22		02	H34	NEXT H35, SPACE 01 AFTER			
23		03		A (30) BLANKS			
24		03		A (07) 'INVOICE'			
25		03		A (18) BLANKS			
26		03		A (06) 'AMOUNT'			
27		03		A (12) BLANKS			
28		03		A (02) 'BY'			
29		03		A (17) BLANKS			

FIGURE 5.6

DECISION TABLE DATA DESCRIPTION

CTL 100	SYSTEM ACCOUNTING			SYSTEM SEGMENT MONTHLY EXPENSE DISTR	PREPARED BY OYE	DATE 8/1/61	PAGE 22 TO PAGE 23
SERIAL	NAME	LEV-EL	TYPE	DESCRIPTION			COZT
30		03		A(17) BLANKS			
31		03		A(02) 'BY'			
32		02	H35	NEXT H36, SPACE 01 AFTER			
33		03		A(30) BLANKS			
34		03		A(06) 'NUMBER'			
35		03		A(06) BLANKS			
36		03		A(06) 'MO DAY'			
37		03		A(22) BLANKS			
38		03		A(07) 'ACCOUNT'			
39		03		A(13) BLANKS			
40		03		A(02) 'BY'			
41		02	H36	SPACE 01 AFTER			
42		03		A(20) ' *** DEPT. NO. '			
43	DEPT	03		9(03)			
44		02	H21	SPACE 01 AFTER, COND F2			
45		03		A(25) ' ** SUB. LEDGER NO. '			
46	GEN.LEDGER	03		9(03)			
47		02	H11	SPACE 01 AFTER, COND F1			
48		03		A(25) ' * SUB. LEDGER NO. '			
49	SUB.LEDGER	03		9(03)			
50		02	DAA	SPACE 01 AFTER			
51		03		A(31) BLANKS			
52	INV.NR	03		9(05)			
53		03		A(06) BLANKS			
54	DATE	03		X(06)			
55		03		A(05) BLANKS			
56	INV.AMT	03		ZZ,ZZ9.99			
57		02	T11	SPACE 02 AFTER, COND F1			

FIGURE 5.6
Report Definition - Data Description (Part 2 of 3)

DECISION TABLE DATA DESCRIPTION

CTL 100	SYSTEM ACCOUNTING			SYSTEM SEGMENT MONTHLY EXPENSE DISTR	PREPARED BY OYE	DATE 8/1/61	PAGE 23 TO PAGE FINAL
SERIAL	NAME	LEVEL	TYPE	DESCRIPTION			COMT
58		03		A(52) BLANKS			
59	TOT.AMT	03		ZZZ,ZZ9.99			
60		03		X(02) ' *'			
61		02	T21	SPACE 02 AFTER, COND F2			
62		03		A(67) BLANKS			
63	AMT.ACT	03		Z,ZZZ,ZZ9.99			
64		02	T31	SPACE 01 AFTER, COND F3			
65		03		A(84) BLANKS			
66	AMT.DEP	03		ZZ,ZZZ,Z9.99			
67		02	TAA	SPACE 03 BEFORE, COND END.REPT			
68		03		A(69) BLANKS			
69		03		A(14) 'FINAL AMOUNT '			
70	FIN.AMT	03		ZZZ,ZZZ,ZZ9.99			
71		02	HB1	NEXT HB2, SKIP 02 BEFORE, SKIP 03 AFTER, COND OF, NF3, COPY H32			
72		02	HB2	NEXT HB3, SPACE 01 AFTER, COPY H33			
73		02	HB3	NEXT HB4, SPACE 01 AFTER, COPY H34			
74		02	HB4	SPACE 01 AFTER, COPY H35			

FIGURE 5.6
Report Definition -- Data Description (Part 2 of 2

APPENDIX

SAMPLE INVENTORY PROBLEM

A simplified inventory problem has been selected for illustrating the use of decision tables. A machine run decision logic is documented from the analyst viewpoint. It ignores practically all of the machine considerations of how the decision logic will be accomplished.

We can think of a procedure as a series of instruction steps in a specific sequence where each step may be conditioned by the assumption that a previous step(s) has been taken. A completely non-procedural documentation of a problem decision logic would be characterized by:

- Arbitrary selection of points for defining inputs and outputs in terms of records (subordinate data associated with identification data).
- No ordering of data within records.
- No ordering of records within files.
- May or may not separate inputs into multiple files for communication benefits.
- Define each element and item of data as output records in terms of input data conditions and transformation logic without recourse to intermediate work area, files, etc.

This would give near maximum flexibility for procedurizing the implementation of the problem statement. It would permit the simulation and comparison of various problem solutions.

The use of decision tables is a compromise between procedural solution statement of a problem and a completely non-procedural statement of a problem decision logic.

The sample inventory problem documentation is procedure oriented in the following ways:

- The files are sequenced.
- The fields within records are sequenced (however, other than for the report, it would make little difference if the fields were not sequenced within records.)
- Table to table sequence is specific.
- Mandatory sequence of actions within a rule is stated.

The documentation is non-procedural within a table in the following ways:

- Rules within a table can be considered in any sequence (the exception is - the ELSE rule must be considered last in a table).
- Conditions within a rule may be tested in any sequence.

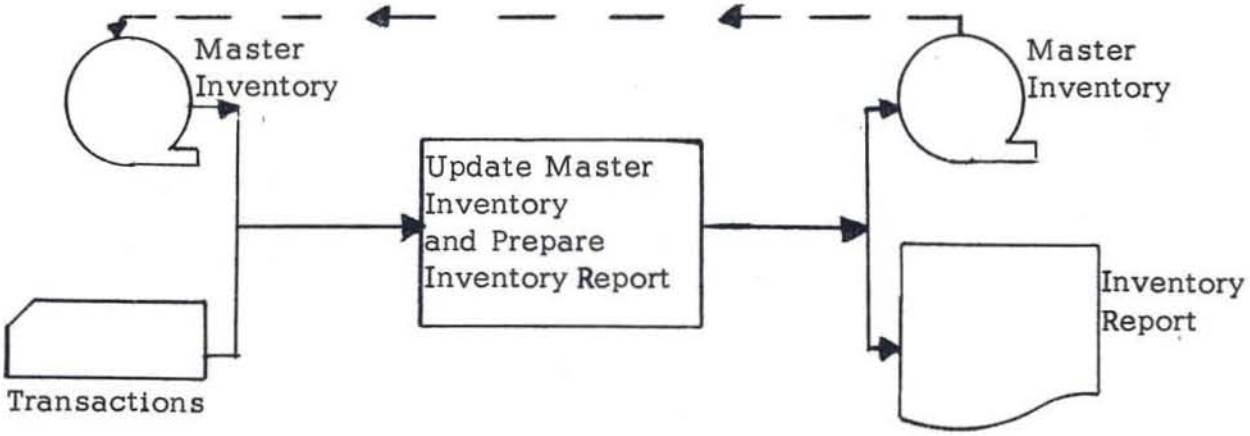
The process chart shows there will be:

Inputs:

Master Inventory - Magnetic Tape
Transactions - Cards

Outputs:

Master Inventory - Magnetic Tape
Report - Printed Report



The sequence of the files are:

FIELDS	SEQUENCE	MASTER	TRANSACTIONS	REPORT
STOCK SYMBOL	MAJOR	X	X	X
LOCATION	INTERMEDIATE	X	X	X
TRANSACTION CODE	MINOR		X	

Other assumptions are:

1. There are no stock symbols with a value of zeros.
2. There is only one master record for a stock symbol and location code.
3. There is a master record for (stock symbol and location code) for each transaction.
4. There may be master records with no activity (no matching transactions.)
5. For any master record there may be multiple transactions of each of the following:
   - Receipts
   - Orders
   - Issues
6. All digits in numeric fields will contain a value from zero through nine.
7. The "quantity on order" field in the master record is always greater than or or equal to "receipt quantity."
8. There will always be one date card and it will be the first card in the transaction file.

INDEX OF INVENTORY PROBLEM DOCUMENTATION

FIGURES

DECISION TABLES

TABLE NR	CLOSED	OPEN	FUNCTION
001		X	Initialize transaction and master inventory files and report headings.
002		X	Position Input Files. Provide control logic for considering other tables. Print Report Lines. Write master file.
003	X		Process Transactions
004	X		When master inventory order quantity equals zero put blanks in order code and order date.
005	X		Determine and prepare management exception instruction.

DATA DESCRIPTION

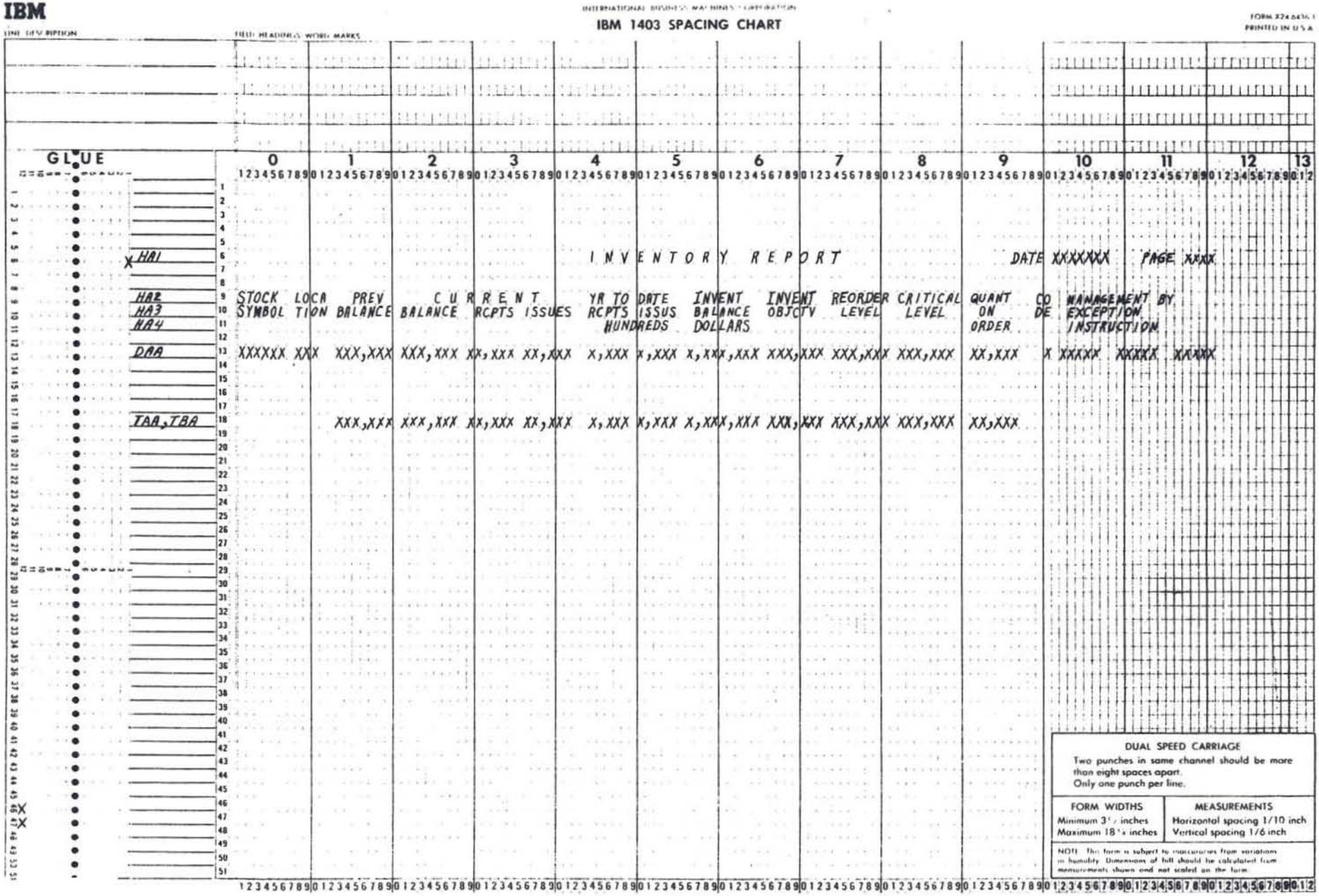


FIGURE A.1
PRINTER SPACING CHART FOR SAMPLE INVENTORY REPORT

I N V E N T O R Y R E P O R T DATE 10APR61 PAGE 10

STOCK SYMBOL	LOCATION	PREV BALANCE	CURRENT BALANCE	CURRENT RCPTS	CURRENT ISSUES	YR TO DATE RCPTS HUNDREDS	YR TO DATE ISSUS HUNDREDS	INVENT BALANCE DOLLARS	INVENT OBJCTV	REORDER LEVEL	CRITICAL LEVEL	QUANT ON ORDER	CODE	MANAGEMENT BY EXCEPTION INSTRUCTION		
121211	A10	600	550	150	200	30	40	6,000	1,000	700	300	0	R		ORDER	450
	A11	80	55	0	25	3	4	5,500	130	100	85	0	R	CRIT	ORDER	75
	A12	100	80	0	20	4	10	8,000	700	400	150	0	R	CRIT	REPTD	10MAR
		780	685	150	245	37	54	19,500	1,830	1,200	535	0				
122220	A10	13,000	10,000	0	3,000	500	750	10,000	20,000	14,000	7,000	0	R		REPTD	2APR
	A11	4,500	3,000	0	1,500	250	400	3,000	10,000	7,000	3,500	0	O	CRIT	ORDED	11MAR
	A12	500	400	0	100	25	40	400	1,000	700	350	0	O		ORDED	20MAR
		18,000	13,400		4,600	775	1,190	13,400	31,000	21,700	10,850					
		18,780	14,085	150	4,845	812	1,244	32,900	32,830	22,900	11,385					

FIGURE A.2
SAMPLE INVENTORY REPORT

BEGIN	TABLE TITLE				SYSTEM SEGMENT WEEKLY REPORT		SYSTEM INVENTORY		PREPARED BY OYE		OPEN X CLOSED		DATE 9-4-61		ELSE		NEXT 002		TABLE 01	
LINE NR	VERB	OPERAND 1	OP	OPERAND2	RULE 1	FREQ 1	RULE	FREQ	RULE	FREQ	RULE	FREQ	RULE	FREQ	RULE	FREQ	RULE	FREQ		
					*	OPERAND2	*	OPERAND2	*	OPERAND2	*	OPERAND2	*	OPERAND2	*	OPERAND2				
10	READ	TRANS			1	X														
11	READ	INV			1	X														
12	MOVE	TRANS DATE	TO	DAT.WA. DATE	2	X														
13	READ	TRANS			3	X														
14	MOVE	INV BAL	TO	PRT.WA PREV.BAL	2	X														
15	DO	BEG.RPT.1			3	X														

*For conditions: OPERATOR
For actions: SEQUENCE

TABLE 001

TABLE TITLE	SYSTEM SEGMENT	SYSTEM	PREPARED	OPEN X	DATE	ELSE	NEXT TABLE	
POSITION FILES-CONTROL LOGIC	WEEKLY REPORT	INV	BY OYE	CLOSED	9-4-61		002	002

LINE NR	VERB	OPERAND 1	OP	OPERAND2	RULE 1 FREQ 2300		RULE 2 FREQ 1000		RULE 3 FREQ 200		RULE 4 FREQ 500		RULE 5 FREQ 1		RULE	FREQ
					*	OPERAND2	*	OPERAND2	*	OPERAND2	*	OPERAND2	*	OPERAND2	*	OPERAND2
01		TRANS STOCK	CMP	INV STOCK	GR		EQ				EQ					
02		TRANS LOC	CMP	INV LOC			EQ				GR					
03		INV	EQ	END		N		N		N		N		Y		
04		TRANS	EQ	END		N		N		Y		N		Y		
10	DO	003					1	X								
11	READ	TRANS					2	X								
12	DO	004			1	X			1	X	1	X				
13	MOVE	CAL.BAL. DOL	TO	PRT.WA BAL.WALU	1	X			1	X	1	X				
14	DO	005			2	X			2	X	2	X				
15	MOVEC	INV	TO	PRT.WA	2	X			2	X	2	X				
16	DO	REPT.1			3	X			3	X	3	X				
17	MOVE	ZEROS	TO	PRT.WA CUR.RCT	4	X			4	X	4	X				
18	"	"	TO	PRT.WA CUR.ISS	4	X			4	X	4	X				
19	WRITE	INV.0	FROM	INV	4	X			4	X	4	X				
20	READ	INV			5	X			5	X	5	X				
21	MOVE	INV BAL	TO	PRT.WA PREV.BAL	6	X			6	X	6	X				
22	DO	END.RPT.1											1	X		
23	STOP												2	X		

*For conditions: OPERATOR CAL.BAL.DOL = INV UNIT.COST * INV BAL
For actions: SEQUENCE

TABLE 003 - Process Transactions
TABLE 004 - Spaces to Ord.Code & Date if Ord.Qty.is zeros
TABLE 005 - Mgt by exception Instr.

TABLE 002

TABLE TITLE	SYSTEM SEGMENT	SYSTEM	PREPARED	OPEN ___	DATE	ELSE	NEXT	TABLE
PROCESS TRANSACTIONS	WEEKLY REPORT	INV.	BY OYE	CLOSED X	9-4-61			003

LINE NR	VERB	OPERAND 1	OP	OPERAND2	RULE 1 FREQ 800		RULE 2 FREQ 125		RULE 3 FREQ 75		RULE FREQ		RULE FREQ		RULE FREQ	
					*	OPERAND2	*	OPERAND2	*	OPERAND2	*	OPERAND2	*	OPERAND2	*	OPERAND2
01		TRANS TR.CODE	EQ			ISSU.T.		RECP.T		ORDR.T						
10	SET	INV BAL	EQ+	TRANS QTY			1	X								
11	↓	" "	EQ-	↓ ↓	1	X										
12		INV ISSU.YTD	EQ+		1	X										
13		PRT.WA CUR.ISS	EQ+		1	X										
14		INV RCPT.YTD	EQ+				1	X								
15		INV ORD.QTY	EQ-				1	X								
16		PRT.WA CUR.REC	EQ+				1	X								
17		INV ORD.QTY	EQ+						1	X						
18		INV ORD.DATE	EQ						1	TRANS DATE						
19		INV ORD.CODE	EQ						1	ORDERED						

*For conditions: OPERATOR
For actions: SEQUENCE

TABLE TITLE	SYSTEM SEGMENT	SYSTEM	PREPARED BY	OPEN / CLOSED	DATE	ELSE	NEXT	TABLE
TEST ORDER QUANTITY	WEEKLY REPT	INV	OYE	CLOSED X	9-4-61			004

LINE NR	VERB	OPERAND 1	OP	OPERAND 2	RULE 1 FREQ 2000		RULE 2 FREQ 3000		RULE FREQ		RULE FREQ		RULE FREQ		RULE FREQ	
					*	OPERAND 2	*	OPERAND 2	*	OPERAND 2	*	OPERAND 2	*	OPERAND 2	*	OPERAND 2
01		INV ORD.QTY	EQ	ZEROS		Y		N								
10	MOVE	SPACES	TO	INV ORD.CODE	1	X										
11	MOVE	SPACES	TO	INV ORD.DATE	1	X										

*For conditions: OPERATOR
For actions: SEQUENCE

TABLE 004

TABLE TITLE	SYSTEM SEGMENT	SYSTEM	PREPARED BY	OPEN / CLOSED	DATE	ELSE		NEXT TABLE
MGT BY EXCEPTION INSTRUCTION	WEEKLY REPT	INV	OYE	OPEN ___ CLOSED X	9-4-61			005

LINE NR	VERB	OPERAND 1	OP	OPERAND2	RULE 1 FREQ 4700 *	OPERAND2	RULE 2 FREQ 60 *	OPERAND2	RULE 3 FREQ 40 *	OPERAND2	RULE 4 FREQ 100 *	OPERAND2	RULE 5 FREQ 60 *	OPERAND2	RULE 6 FREQ 30 *	OPERAND2	RULE 7 FREQ 10 *	OPERAND2
01		INV BAL	CMP	INV CRIT LVL			GR		GR		GR		LE		LE		LE	
02		INV BAL	CMP	INV REORD LVL	GR		LE		LE		LE							
03		INV ORD.CODE	EQ				ORDERED		REPTED		NONE		ORDERED		REPTED		NONE	
10	SET	PRT.WA CRIT	EQ			SPACES		SPACES		SPACES		SPACES		" CRIT"		" CRIT"		" CRIT"
11	↓	PRT.WA ORD RPT	↓			SPACES		"ORDED"		"REPTD"		"ORDED"		"ORDED"		"REPTD"		"ORDED"
12	↓	PRT.WA QTY DAT	↓			SPACES		INV ORD DATE		INV ORD DATE		CAL. ORD. QTY		INV ORD. DATE		INV ORD. DATE		CAL. ORD. QTY
13	↓	INV ORD. DATE	↓	DAT. WA DATE								X						X
14	↓	INV ORD. QTY	↓	CAL. ORD. QTY								X						X
15	↓	INV ORD. CODE	↓	REPTED								X						X

*For conditions: OPERATOR
For actions: SEQUENCE

CAL. ORD. QTY = INV OBJ - INV BAL - INV ORD. QTY

TABLE 005

DECISION TABLE DATA DESCRIPTION

CTL	SYSTEM	SYSTEM SEGMENT	PREPARED BY	DATE	PAGE 1
001	INVENTORY	WEEKLY REPORT	OYE	9-4-61	TO PAGE 2

SERIAL	NAME	LEVEL	TYPE	DESCRIPTION	CONT
01	TRANS	01	RECORD		
02	TR.CODE	02		99	
03	DATE.T	03	COND	'00'	
04	RECPT.T	03	COND	'30'	
05	ISSU.T	03	COND	'40'	
06	ORD.T	03	COND	'20'	
07	STOCK	02		X (6)	
08	LOC	02		X (3)	
09	QTY	02		9 (4)	
10	DATE	02			
11	DAY.MON	03		X (2)	
12	MONTH	03		X (3)	
13		02		A (60) BLANKS	
14	DATE	01	REDEF	TRANS	
15	TR.CODE	02		99	
16	STOCK	02		X(6)	
17	DATE	02			
18	DAY.MON	03		XX	
19	MONTH	03		AAA	
20	YEAR	03		XX	
21		02		A (65) BLANKS	

DECISION TABLE DATA DESCRIPTION

CTL	SYSTEM	SYSTEM SEGMENT	PREPARED BY	DATE	PAGE
002	INVENTORY	WEEKLY REPORT	OYE	9-4-61	2 TO PAGE 3

SERIAL	NAME	LEVEL	TYPE	DESCRIPTION	CONT
01	INV	01	RECORD		
02	STOCK	02		X (6)	
03	LOC	02		X (3)	
04	BAL	02		9 (6)	
05	OBJ	02		9 (6)	
06	REORD.LVL	02		9 (6)	
07	CRIT.LVL	02		9 (6)	
08	RCPT.YTD	02		9 (6)	
09	ISSU.YTD	02		9 (6)	
10	ORD.QTY	02		9 (5)	
11	ORD.CODE	02		A	
12	REPTED	03	COND	'R'	
13	ORDERED	03	COND	'O'	
14	NONE	03	COND	SPACE	
15	ORD.DATE	02			
16	DA.MON	03		XX	
17	MONTH	03		XXX	
18	UNIT.COST	02		99V999	
19	INV.O	01	COPY	INV	

SPACES = BLANKS

DECISION TABLE DATA DESCRIPTION

CTL 003	SYSTEM INVENTORY			SYSTEM SEGMENT WEEKLY REPORT	PREPARED BY OYE	DATE 9-4-61	PAGE 3 TO PAGE 4
SERIAL	NAME	LEVEL	TYPE	DESCRIPTION			CONT
01	PRT.WA	01	WORK				
02	STOCK	02		X (6)			
03	LOC	02		X (3)			
04	PREV.BAL	02		9 (6)			
05	BAL	02		9 (6)			
06	CUR.RCT	02		9 (5) ZEROS			
07	CUR.ISS	02		9 (5) ZEROS			
08	RCPT.YTD	02		9 (6)			
09	ISSU.YTD	02		9 (6)			
10	BAL.VALU	02		9 (8) V9 (3)			
11	OBJ	02		9 (6)			
12	REORD.LVL	02		9 (6)			
13	CRIT.LVL	02		9 (6)			
14	ORD.QTY	02		9 (5)			
15	ORD.CODE	02		A			
16	MGT.INS	02					
17	CRIT	03		A (5)			
18	ORD.RPT	03		A (5)			
19	QTY.DAT	03		X (5)			

DECISION TABLE DATA DESCRIPTION

CTL 010	SYSTEM INVENTORY			SYSTEM SEGMENT WEEKLY REPORT	PREPARED BY OYE	DATE 9-4-61	PAGE 4 TO PAGE 4.1
SERIAL	NAME	LEVEL	TYPE	DESCRIPTION			CONT
01	WKLY.INV	01	REPT.1				
02	PRT.WA	02	SOURCE				
03	LOC	03	F1				
04	STOCK	03	F2				
05		02	HA1	NEXT HA2, SPACE 03 AFTER, SKIP 01 BEFORE, COND OF			
06		03		A (43) SPACES			
07		03		A (31) 'I N V E N T O R Y R E P O R T'			
08		03		A (21) SPACES			
09		03		A (5) 'DATE '			
10	DATE	03					
11	DAY.MON	04		99			
12	MONTH	04		AAA			
13	YEAR	04		99			
14		03		A (9) ' PAGE '			
15	PAGE	03		9 (4)			
16		02	HA2	NEXT HA3, SPACE 01 AFTER			
17		03		A (43) 'STOCK LOCA PREV C U R R E N T			
18		03		A (46) 'YR TO DATE INVENT INVENT REORDER CRITICAL'			
19		03		A (26) ' QUANT CO MANAGEMENT BY'			
20		02	HA3	NEXT HA4,SPACE 01 AFTER			
21		03		A (43) 'SYMROL TION BALANCE BALANCE RCPTS ISSUES			
22		03		A (44) 'RCPTS ISSUS BALANCE OBJCTV LEVEL LEVEL'			
23		03		A (24) ' ON DE EXCEPTION'			
24		02	HA4	SPACE 02 AFTER			
25		03		A (45) SPACES			
26		03		A (18) 'HUNDREDS DOLLARS'			
27		03		A (27) SPACES			
28		03		A (23) 'ORDER INSTRUCTION'			

SPACES = BLANKS

DECISION TABLE DATA DESCRIPTION

CTL 010	SYSTEM INVENTORY			SYSTEM SEGMENT WEEKLY REPORT	PREPARED BY OYE	DATE 9-4-61	PAGE 4.1 TO PAGE 5
SERIAL	NAME	LEVEL	TYPE	DESCRIPTION			CTL
40	DETAIL	02	DAA	SPACE 01 AFTER			
41	STOCK	03		X (6) GROUP SUPPRESS			
42		03		A (1) SPACE			
43	LOC	03		X (3)			
44		03		A (2) SPACES			
45	PREV.BAL	03		ZZZ,ZZ9			
46		03		A (1) SPACE			
47	BAL	03		ZZZ, ZZ9			
48		03		A (1) SPACE			
49	CUR.RCT	03		ZZ,ZZ9			
50		03		A (1) SPACE			
51	CUR.ISS	03		ZZ,ZZ9			
52		03		A (2) SPACES			
53	RCPT.YTP	03		Z,ZZ9PP			
54		03		A (1) SPACE			
55	ISSU.YTD	03		Z,ZZ9PP			
56		03		A (1) SPACE			
57	BAL.VALU	03		Z,ZZZ,ZZ9V			
58		03		A (1) SPACE			
59	OBJ	03		ZZZ,ZZ9			
60		03		A (1) SPACE			
61	REOR.LVL	03		ZZZ,ZZ9			
62		03		A (1) SPACE			
63	CRIT.LVL	03		ZZZ,ZZ9			
64		03		A (2) SPACES			
65	ORD.QTY	03		ZZ,ZZ9			
66		03		A (3) SPACES			
67	ORD.CODE	03		A (1)			

SPACES = BLANKS

DECISION TABLE DATA DESCRIPTION

CTL 010	SYSTEM INVENTORY			SYSTEM SEGMENT WEEKLY REPORT	PREPARED BY OYE	DATE 9-4-61	PAGE 5 TO PAGE 6
SERIAL	NAME	LEV-EL	TYPE	DESCRIPTION			CONT
68		03		A (1) SPACE			
69	MGT.INS	03					
70	CRIT	04		A (5)			
71		04		A (2) SPACES			
72	ORD.RPT	04		A (5)			
73		04		A (2) SPACES			
74	QTY.DAT	04		X (5)			
75	STK.TOTAL	02	TAA	SPACE 01 BEFORE, SPACE 02 AFTER, COND F2			
76		03		A (12) SPACES			
77	PREV.BAL	03		ZZZ,ZZ9			
78		03		A (1) SPACE			
79	BAL	03		ZZZ,ZZ9			
80		03		A (1) SPACE			
81	CUR.RCT	03		ZZ,ZZ9			
82		03		A (1) SPACE			
83	CUR.ISS	03		ZZ,ZZ9			
84		03		A (2) SPACES			
85	RCPT.YTD	03		Z,ZZ9PP			
86		03		A (1) SPACE			
87	ISSU.YTD	03		Z,ZZ9PP			
88		03		A (1) SPACE			
89	BAL.VALU	03		Z,ZZZ,ZZ9V			
90		03		A (1) SPACE			
91	OBJ	03		ZZZ,ZZ9			
92		03		A (1) SPACE			
93	REORD.LVL	03		ZZZ,ZZ9			

SPACES = BLANKS

DECISION TABLE DATA DESCRIPTION

CTL 011	SYSTEM INVENTORY			SYSTEM SEGMENT WEEKLY REPT	PREPARED BY OYE	DATE 9/4/61	PAGE 6 TO PAGE FINAL

SERIAL	NAME	LEVEL	TYPE	DESCRIPTION	CONT
01		03		A(1) SPACE	
02	CRIT.LVL	03		ZZZ,ZZ9	
03		03		A(2) SPACES	
04	ORD.QTY	03		ZZ,ZZ9	
05	REPT.TOT	02	TBA	SPACE 02 BEFORE, COND END.REPT.1, COPY TAA	
06	DAT.WA	01	WORK		
07	DATE	02			
08	DAY.MON	03		99	
09	MONTH	03		AAA	
10	YEAR	03		99	
11	CAL.BAL.DOL	01	EXPRES	9(8)V9(3) INV UNIT.COST * INV BAL	
12	CAL.ORD.QTY	01	EXPRES	9(6) INV OBJ - INV BAL - INV ORD.QTY	

SPACES = BLANKS