SYSTEMS ENGINEERING SERVICES

PRELIMINARY MANUAL

7080

DECISION TABLE SYSTEM

April 10, 1962
Technical Report No. 2D1

Thomas B. Glans
Burton Grad

INTERNATIONAL BUSINESS MACHINES CORPORATION

White Plains, New York

IBM 7080 DECISION TABLE SYSTEM

Preface

Of the various activities that go into setting up a data processing procedure for a computer, the hardest is obtaining a precise definition of exactly what is to be done under all combinations of circumstances. The conditions under which each action is to be performed must be enumerated, and every step must be specified in detail. The sequence in which operations are to be executed must be precisely stated, and the exceptions to normal processing must be identified.

Logical operations have generally been described with block diagrams, which are a form of man-to-man communication and documentation. Unfortunately they can become confusing themselves in complex situations; and a block diagram is generally not suitable for communication with the machines, since a computer will accept as input only alphameric information, not pictures or diagrams.

Many languages have been developed for communication with the machine in forms which more closely resemble the terminology of English or of mathematics. These programming systems such as FARGO, Reports Generators and FORTRAN are oriented toward some area of data processing for which they are primarily intended such as input-output, or mathematical equations. Decision tables now offer a means for expressing complex decision logic in connection with these languages, or independent of them when the essence of the problem is decision logic.

Logical equations (Boolean Algebra) are sometimes used, but they do not display relationships in as graphical a form as one might wish, and furthermore most systems engineers and procedures analysts do not find them to be a comfortable form of expression.

Decision tables are proposed as holding promise of meeting the various needs of a computer language. Decision tables provide a descriptive representation of complex decision procedures in a way that is easy to visualize and follow through. They show alternatives and exceptions in a much more explicit way than other languages, while presenting relationships among variables clearly. They show the sequences of conditions and actions in an unambiguous manner. The language of decision tables can therefore be used equally effectively for system analysis, procedure design, and computer coding. A computer procedure written as a set of decision tables is, to a large extent, its own documentation.

There is a growing body of experience to indicate that these claims are justified. Those who have used decision tables say that programming is much faster and that program checkout time is significantly reduced. The use of tables leads to greater accuracy and completeness in problem formulation. Program maintenance is simpler. A program written in decision table form is indeed a powerful communication and documentation device, since it performs the dual functions of man-to-man and man-to-machine communication.

For these reasons the employment of decision tables for problem analysis, logic documentation and computer programming should be given serious consideration. User experience will help in the evaluation of the concept, and will assist in improving its implementation in terms of processor versatility in translating from decision tables to computer instructions.

A word about this teaching text. It is directed primarily at people who will be experimenting with the language. Since many such people may be stronger in systems work than in programming, the presentation discusses programming matters more fully than an experienced programmer requires. However, this redundancy should not detract from the technical content of the manual.

Chapter 1 introduces the concept of decision tables and discusses the preparation of decision tables for machine processing.

Chapter 2 shows how these concepts are implemented in the 7080 Decision Table System and presents the essential elements of the system and its language. This chapter utilizes a graded series of examples -- the first ones are quite simple, but become increasingly complex as further concepts are introduced and earlier concepts are tied together. There are review questions and answers after each example.

Chapter 3 contains a full-scale problem worked out with the 7080 Decision Table System. It shows how the concepts can be applied to a specific situation.

Chapter 4 is a reference section, describing the details of the language elements and their relationships.

It is recommended that the person without extensive programming background study the entire manual carefully, concentrating on the examples and questions in Chapter 2. The experienced programmer will probably want to read Chapters 1 and 2 and concentrate on Chapters 4 and 3.

Two final comments:

(1) Because of time limitations and in light of the experimental purpose of the language, certain functions that might have been included were left out. However, the system will nevertheless be useful in evaluating the concept of tabular programming for a large-scale computer, and along with the evaluation of the overall concept will come suggestions for improvements in the details. All such comments and suggestions will be appreciated.

(2) It should be understood that IBM makes no commitment to maintain or improve this language. Although every reasonable attempt has

been made to pretest the processor and to examine various implications of the language, the accuracy or efficiency of the processor is not guaranteed, nor are the correctness of results obtained with its use.

Thomas B. Glans

Burton Grad

April 1, 1962

ACKNOWLEDGMENTS

The development of the 7080 Decision Table System has been a joint effort between The Boeing Company and IBM. Messrs. Donald Higgins, Harry Hicks and James O'Malley of Boeing participated in specifying the language. In addition, H. Hicks did an excellent job in writing a large portion of the processor. J. O'Malley and a member of his staff, produced the fine sample problem of Chapter 3 which was also used in checking out the processor.

Messrs. R. N. Barnes, O. Y. Evans and H. W. Stroeve of the IBM Western Regional Office and C. A. Mabee of the Seattle Branch Office also participated in the language design effort. In addition they provided valuable consulting and administrative assistance throughout the implementation stage of the project.

Mr. H. J. Cash of DP Technical Publications did the major portion of the writing of this manual. Mr. R. J. Schatz of DP Systems Publications set up and printed the manual.

TABLE OF CONTENTS

Page

Chapter 4 - 7080 Decision Table System Specification

Page

Chapter 4 (cont'd.)

Appendix

CHAPTER 1

An Introduction to the 7080 Decision Table System

Some orderly arrangement of data is necessary for the logical solution of any complex problem. At first blush, for example, the puzzle below does not lend itself to an easy solution:

> Andy disliked the catcher. Ed's sister was engaged to the second baseman. The center fielder was taller than the right fielder. Harry and the third baseman lived in the same building. Paul and Allen each won $20 from the pitcher at pinochle. Ed and the outfielders played poker during their free time. The pitcher's wife was the third baseman's sister. All the battery and infield, except Allen, Harry and Andy, are shorter than Sam. Paul, Andy and the shortstop lost $50 each at the race track. Paul, Harry, Bill and the catcher took a trouncing from the second baseman at pool. Sam was undergoing a divorce suit. The catcher and the third baseman each had two children. Ed, Paul, Jerry, the right fielder and the center fielder were bachelors. The others were married. The shortstop, the third baseman and Bill each cleaned up $100 betting on the fight. One of the outfielders was either Mike or Andy. Jerry was taller than Bill. Mike was shorter than Bill. Each of them was heavier than the third baseman.

With these facts determine the names of the men playing each position on the baseball team.

It is, however, far less complex than it appears when the problem is broken down into a series of steps. What are the facts which determine the solution?

Obviously (1) the names of the players and (2) their position on the team.

	C	P	1st	2nd	3rd	SS	LF	RF	CF
Andy	N								
Ed				N					
Harry					N				
Paul		N							
Allen		N							
Bill									
Sam									
Jerry									
Mike									

The first sentence of the puzzle states that Andy is not the catcher, and the second states that Ed is not the second baseman. Instead, however, of retaining mentally all such facts, they can be itemized as above until the process of elimination takes care of all negatives and the remaining entries are all "Yes". The solution is stated below:

	P	C	1st	2nd	3rd	SS	LF	RF	CF
Andy	N	N	N	N	Y	N	N	N	N
Ed	N	N	N	N	N	Y	N	N	N
Harry	Y	N	N	N	N	N	N	N	N
Paul	N	N	Y	N	N	N	N	N	N
Allen	N	Y	N	N	N	N	N	N	N
Sam	N	N	N	N	N	N	Y	N	N
Bill	N	N	N	N	N	N	N	N	Y
Jerry	N	N	N	Y	N	N	N	N	N
Mike	N	N	N	N	N	N	N	Y	N

At the heart of any data processing problem is a set of decisions the computer must make regarding the data contained on each record and the actions to be taken as a result of these decisions. The decisions may be regarded as a set of conditions and when a given set of conditions is satisfied then a certain action or set of actions is to be taken.

In a payroll application, for example, if a salaried employee has not been absent and if he has not worked overtime, then his salary is computed with normal deductions. "Compute salary with normal deductions," in itself, constitutes a set of actions. Of course there are many more if considerations in any payroll application, and when a given set of conditions is not satisfied, another series of actions must be taken.

A decision table defines all conditions and separates them from all actions. Further, it relates a given condition to the appropriate actions in an entry. A series of conditions that must result in a given series of actions constitutes a rule. Alternative conditions that result in other actions constitute other rules.

Basically then, a decision table:

- relates a series of conditions and actions
- aligns alternative rules side by side

The Structure of a Table

To begin to see what decision tables are all about, examine the simple table of Figure 1.

	Rule 1	Rule 2	Rule 30
Age	25-34	25-34	65 or over
Health	Excellent	Excellent	Poor
Section of Country	East	West	West
Rate/1000	1.57	1.72	5.92
Policy Limit	200,000	200,000	20,000

Figure 1. An example of a decision table

The first decision rule (columns 1 and 2) can be paraphrased: If age is greater than or equal to 25 and 34 or less, and health is excellent, and section of country is East, then set rate per thousand to 1.57 and set policy limit to 200,000. The underlined words are implied by the table layout. The other rules are alternatives to this one, so that logically, only one rule can be satisfied in a single pass through this decision table.

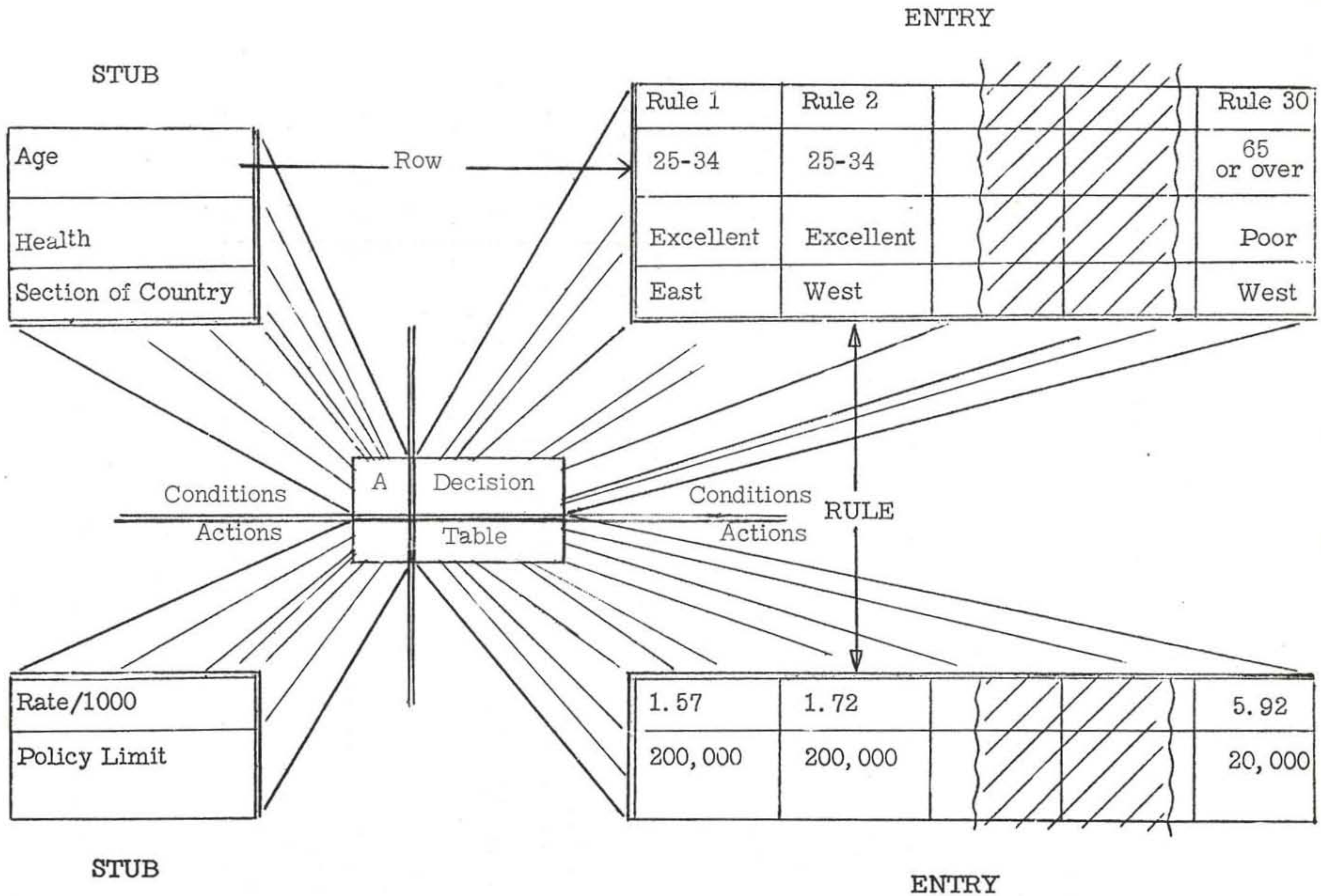


Figure 2. Exploded view of the table of Figure 1.

The information in Figure 1 is shown in an exploded view in Figure 2, to show more clearly the parts of a table and the terms that are used to describe them.

The double horizontal and vertical lines serve as demarcation: CONDITIONS are shown above the horizontal double line, ACTIONS below; the STUB is to the left of the vertical double line, ENTRIES are to the right. Each vertical combination of conditions and actions is called a decision RULE.

Tables may also be used in a slightly different way to state decision logic, as shown in Figure 3.

Rule No.	1	2	3	4
Credit Limit is OK	Y	N	N	N
Pay Experience is Favorable		Y	N	N
Special Clearance is Obtained			Y	N
Approve Order	X	X	X	
Return Order to Sales				X

Figure 3. A limited entry table

The first rule (the stub and column 1) is read: _If_ credit limit is

OK, <u>then</u> approve order. Again, the underlined words are implied by the form. Note that the form of both conditions and actions is different. In a limited entry condition or action, the entire condition or action must be written in the stub; the entry is <u>limited</u> to asserting, reversing, or ignoring a condition or executing an action. In contrast the <u>extended entry</u> form (as in Figure 1) has part of the condition or action <u>extended</u> into the entry. Both forms may be used within one table, but any one condition or action row must be of just one form.

Note that the basic concept of a single rule in a table is based on the "if... then" relationship in a straightforward manner. <u>If</u> A = B <u>and</u> C is greater than 5 <u>and</u> . . . <u>then</u> assign the value 7 to X <u>and</u> GO TO Table 10. The alternative rules are explicit; if the conditions in rule 1 are not met, then try rule 2. If none of the rules succeeds, then the implication is that something is wrong. Now if all significant possibilities are exhausted, it may be desirable to indicate "Go ahead anyhow with a special routine." This is indicated by placing "ELSE" in the last rule; otherwise, provisions must be made for an automatic error detection.

The skeletonized table in Figure 4 is directly related to the block diagram in Figure 5, which shows schematically the way in which a table is conceptually "executed."

Cond 1	Y	Y	Y		ELSE
Cond 2	Y		N		
Cond 3	Y	N	Y		
Action 1	X	X			X
Action 2	X	X	X		X

Figure 4. A decision table of the logic of Figure 5

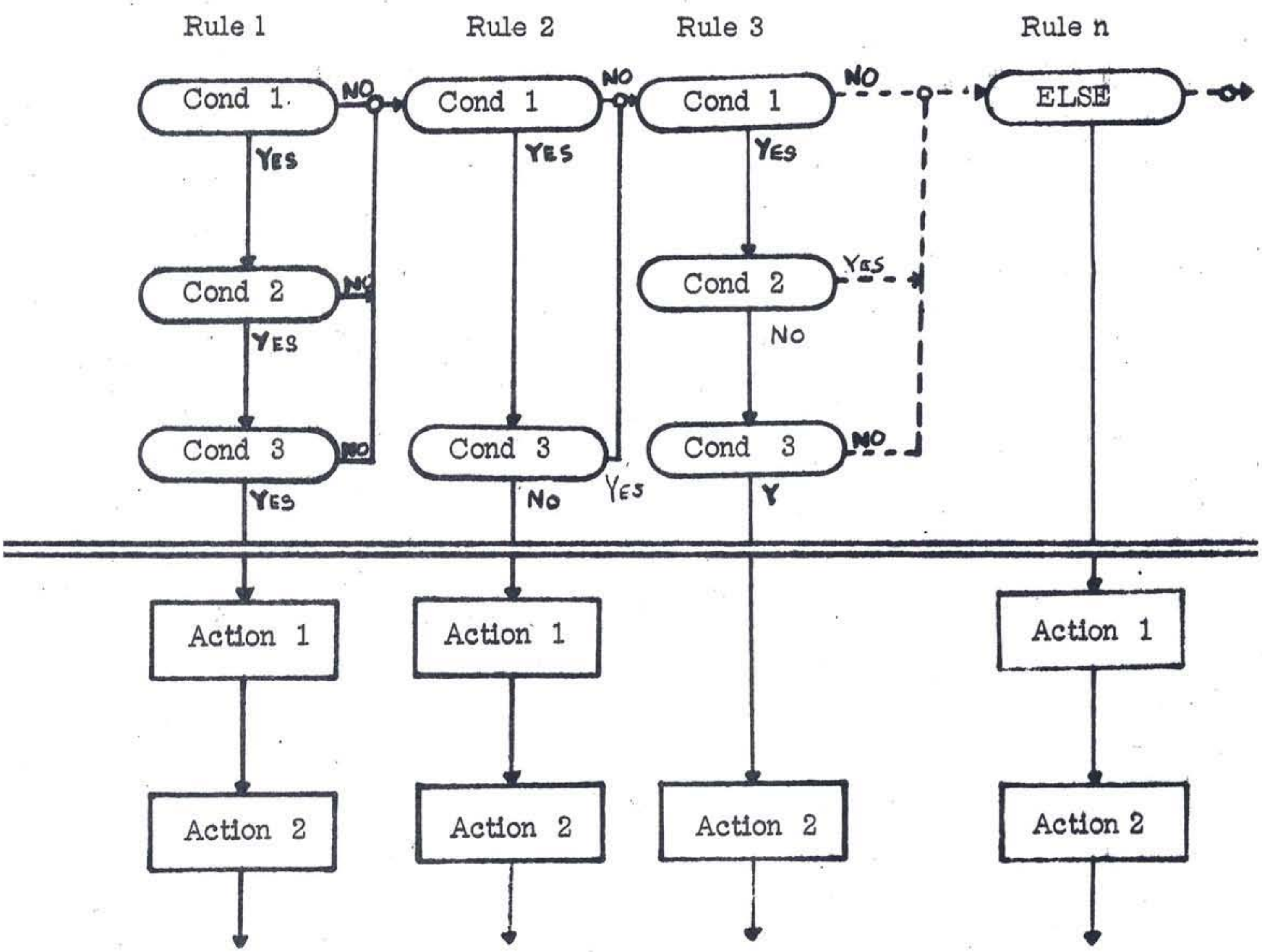


Figure 5. Schematic representation of the sequence of tests and actions in executing a table.

7080 Decision Table Processor

The use of decision tables as a language for prescribing procedures to a computer has been discussed. It should be noted that a table cannot be directly executed by a computer. It is necessary first to translate the decision tables into basic machine instructions. The set of tables that define the processing to be done by the computer makes up the source program; this is translated into an object program--consisting of actual machine instructions--by a processor which is, itself, a special type of program. It is the object program that finally directs the computer to carry out the desired processing of data.

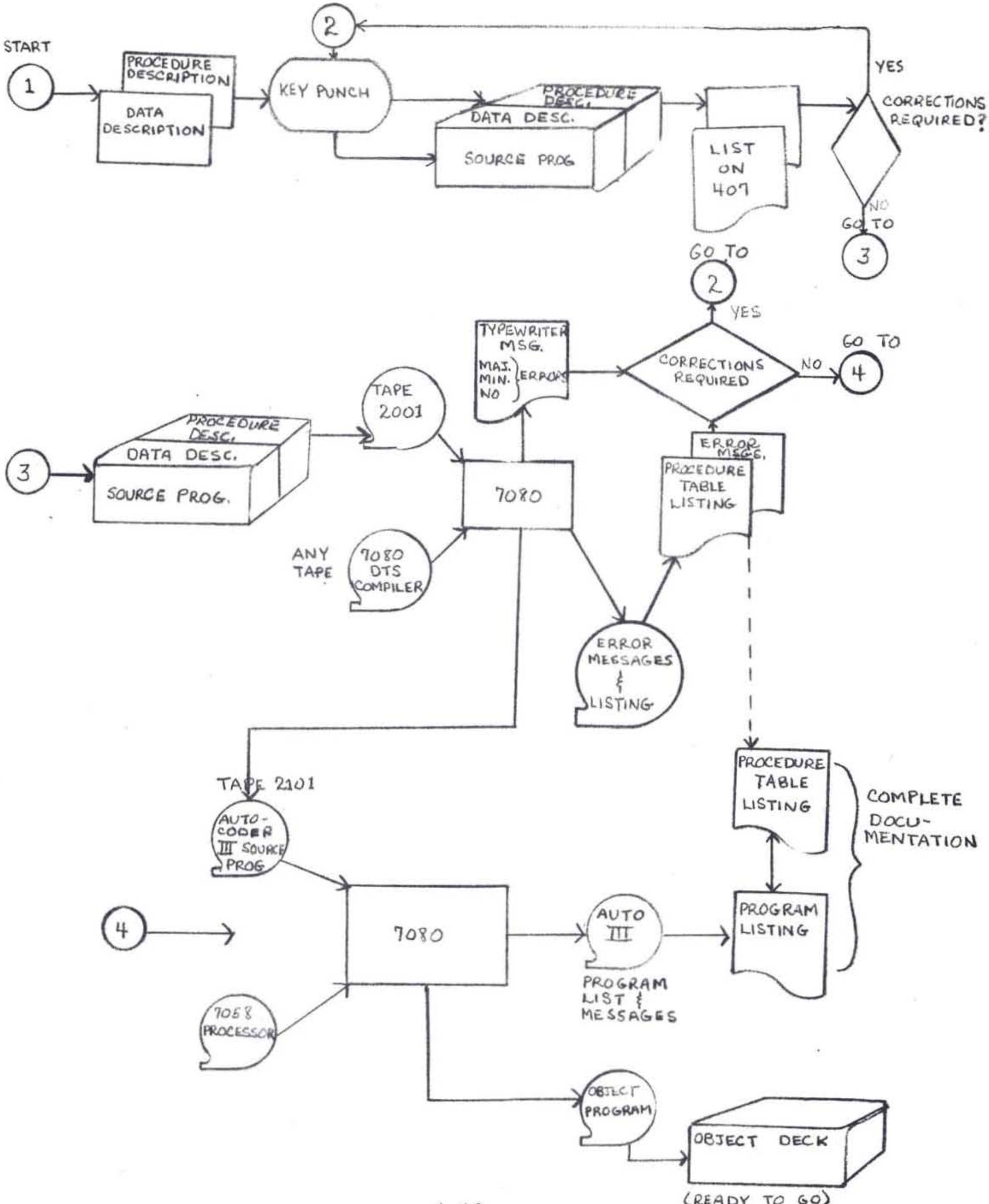
The procedure for going from decision tables written on paper to final problem results is shown in Figure 6.

(1) The first step is the one that is the primary subject of this text: planning the procedure and writing it as a set of decision tables.

(2) Cards are punched and verified, producing the source program card deck (these cards are converted to magnetic tape). The processor program, mentioned above, is on tape.

(3) The processor program goes into the computer and directs the translation of the source program into an object program consisting of computer instructions. The object program is written on tape. There are also two listings produced by the processor, giving information about the source and object program.

SYSTEM DIAGRAM



The processing is in fact done in two steps: one to go from source program to an Autocoder program (Autocoder is another computer language, at a higher level than machine language), and a second to go from Autocoder to machine instructions. Conceptually, however, the two parts are not essentially distinct.

(4) The object program is now available on magnetic tape; it has not yet been executed.

(5) The object program may now be loaded into the computer to do the data processing specified by the original decision tables. It is only at this stage that data is processed. Other computer components are used as required by the program.

The machine that executes the object program need not be the same one which processed the source program. Within certain limits, the two machines need not even have the same features. The machine configuration for each of these jobs is shown in the Appendix.

At this point the basic ideas and use of decision tables have been presented in rather broad outline. The next step is to learn how these ideas are implemented in the 7080 Decision Table System.

PAGE	LINE	C	ACTION OPERATOR	NAME 1	OP	NAME 2	RULE 1	RULE 2	RULE 3
	01	A		TYPECODE	EQ		1	2	3
	02	A							
	03	A	SET	MODEL	EQ		'CONVRTBL'	'HARDTOP'	'STATNWGN'
	04	A							

Situation 1

CHAPTER 2

Using The 7080 Decision Table System

The basic ideas of the 7080 Decision Table System are relatively easy to learn. This chapter presents these ideas in the framework of a series of problem situations, each chosen to illustrate a few new concepts. At the end of the discussion of each situation there are review questions, with answers on the back of the same page, to permit the reader to check his understanding of the material before proceeding.

This chapter does not discuss all the features of the language. For example, it omits some of the details of alternative ways of doing things. The complete information appears in Chapter 4.

For a first example, consider a situation in which it is necessary to convert an automobile body classification from a code to a readable abbreviation.

The table on the opposite page can be interpreted fairly readily. In English:

- If the type code of the car body is 1, then the model is a convertible.
- If the type code of the car body is 2, then the model is a hardtop.
- If the type code of the car body is 3, then the model is a station wagon.

PAGE	LINE	C	ACTION OPERATOR	NAME 1	OP	NAME 2	RULE 1	RULE 2	RULE 3
	01	A		TYPECODE	EQ		1	2	3
	02	A							
	03	A	SET	MODEL	EQ		'CONVRTBL'	'HARDTOP'	'STATNWGN'
	04	A							

Situation 1

In this table, there is one condition row (above the double horizontal line): The action to be carried out depends on whether TYPECODE = 1, 2, or 3. There is also one action row (below the double horizontal line): set MODEL equal to CONVRTBL, HARDTOP, or STATNWGN, depending on whether TYPECODE = 1, 2, or 3 respectively. This illustrates the basic idea of a table, that conditions and actions are always related: if a specified condition (or, more generally, a set of conditions) is satisfied, then the corresponding action (or actions) is executed. A vertical column containing one or more conditions above the double horizontal line and one or more actions below, is called a decision rule. Thus, Rule 1 reads:

If TYPECODE = 1, then SET MODEL EQ 'CONVRTBL'.

TYPECODE is a name for the data field containing the type code; similarly, MODEL is the name of the field that will contain the actual abbreviation for the body type. TYPECODE and MODEL are field names; they represent variables consisting in one case of a one-digit code and, in the other, of an eight letter abbreviation. Furthermore, TYPECODE and MODEL are names of operands -- that is, information fields on which operations of one sort or another are performed.

In the action part of the table, SET...EQ... is an action operator which describes operations to be performed when the appropriate conditions are satisfied. In this case, the action is to transfer one of the abbreviations, 'CONVRTBL', 'HARDTOP', or 'STATNWGN' to the field named MODEL, depending on which condition is satisfied.

PAGE	LINE	CC	ACTION OPERATOR	NAME 1	OP	NAME 2	RULE 1	RULE 2	RULE 3
	01	A		TYPECODE	EQ		1	2	3
	02	A							
	03	A	SET	MODEL	EQ		'CONVRTBL'	'HARDTOP'	'STATNWGN'
	04	A							

Situation 1

The words 'CONVRTBL', 'HARDTOP', and 'STATNWGN' are <u>not</u> field names, but rather are called <u>literals</u>, as indicated by the single quotation marks. The distinction between a field name and a literal is most important. Without the quotation marks, Rule 1 would read:

> If TYPECODE = 1, transfer the current value of the data field named CONVRTBL to the data field named MODEL.

This clearly is not the same thing at all as moving the letters CONVRTBL themselves to the data field named MODEL.

In fact, in this particular table everything to the right of the vertical double line, i.e., the entry part, is a literal: the 2, for instance, refers to the <u>literal</u> value 2, not to a field name 2. Short numeric literals are not written with quotation marks since they could not be confused with field names -- which are not allowed to begin with a numeric digit. Alphameric literals, on the other hand, must be enclosed in quotes to distinguish them from the names of fields.

If the reader can correctly answer the following questions, he is ready to proceed; otherwise, some review will help to understand the later material more quickly. Answers are given on the back of this page.

QUESTIONS

1. List the field names in Situation 1.
2. List the literals.
3. What is the difference between a field name and a literal?
4. State Rule 3 in words.
5. What is an action operator?

Answers - Situation 1

1. TYPECODE, MODEL.
2. 1, 2, 3, 'CONVRTBL', 'HARDTOP', 'STATNWGN'.
3. A data field is referred to by <u>name</u>, whereas a literal has only its actual value, and does not refer to anything else.
4. If TYPECODE = 3, set the field named MODEL equal to the letters STATNWGN. Stated otherwise: If TYPECODE = 3, transfer the literal value 'STATNWGN' to the field named MODEL.
5. A command to perform some action on one or more operands.

PAGE	LINE	C	ACTION OPERATOR	NAME 1	OP	NAME 2	RULE 1	RULE 2
	01	A		PAID	EQ		'N'	'G'
	02	A						
	03	A	SET	CUSTNAME	EQ		INPUTNAME	INPUTNAME
	04	A	SET	PAIDAMOUNT	EQ		PAIDNET	PAIDGROSS
	05	A	PUT				RECEIPT	RECEIPT
	06	A	GET				BILLSTUB	BILLSTUB
	07	A	GO TO				TAB0006	TAB0006
	08	A						

Situation 2

Situation 2 involves a utility billing procedure. The bill stub that a customer returns with his payment has punched in it a net amount and a gross amount. Before the bill stub goes onto a card image tape and into the computer system, the letter N or G is punched into the card to indicate which amount was paid; it is then unnecessary to punch the amount of the payment.

In English, this table reads:

Rule 1: If the card field named PAID contains 'N' (for net), transfer the customer's name from the input field named INPUTNAME to the output field named CUSTNAME, transfer the field named PAIDNET to PAIDAMOUNT, write a receipt record, read another bill stub record, and finally GO TO table 0006.

Rule 2: If the PAID field contains 'G' (for gross), carry out the same actions as for Rule 1, except transfer the gross amount from PAIDGROSS to PAIDAMOUNT as spelled out in action row 2.

The single condition row in this example is quite similar to that in the previous example, with just one difference: since the literals are alphabetic they must be enclosed in quotes.

The SET... EQ ... action operators in the first two action rows are the same as before. The PUT operator, however, is new; it is the terminology of the 7080 Decision Table System which means to make data available for the writing of an output record.

PAGE	LINE	C	ACTION OPERATOR	NAME 1	OP	NAME 2	RULE 1	RULE 2
	01	A		PAID	EQ		'N'	'G'
	02	A						
	03	A	SET	CUSTNAME	EQ		INPUT NAME	INPUT NAME
	04	A	SET	PAIDAMOUNT	EQ		PAIDNET	PAIDGROSS
	05	A	PUT				RECEIPT	RECEIPT
	06	A	GET				BILLSTUB	BILLSTUB
	07	A	GO TO				TAB0006	TAB0006
	08	A						

Situation 2

PUT is part of the specified language of the 7080 DTS; but this example and the previous example have used named operands which are not part of the basic language of the system, but rather names chosen by the problem analyst. These names are for variable data fields, records and files. A variable data field, of course, contains a value. A record is a collection of fields, and a file is a collection of one or more records associated with an input or output device.

In this situation, PUT RECEIPT specifies making a record from the file named RECEIPT available for writing -- here the data is the customer's name and the amount paid. To summarize, the PUT action operator requires a file name for its operand and it calls for the outputting of a record to this file.

Such data records normally consist of fields which are of known length and arrangement. Since input and output areas, work areas, areas for storing constants, etc., must be reserved in core storage for all records, their length and arrangement must be specified. These areas will then automatically be assigned to memory locations in the order in which they are defined in the program. The specification of these definitions is made on a 7080 Decision Table Name Description Sheet. The writing of procedures and operand descriptions are separate programmer's jobs and every program requires both. Data description, being another consideration, will be explained in another example; so

PAGE	LINE	C	ACTION OPERATOR	NAME 1	OP	NAME 2	RULE 1	RULE 2
	01	A		PAID	EQ		'N'	'G'
	02	A						
	03	A	SET	CUSTNAME	EQ		INPUT NAME	INPUT NAME
	04	A	SET	PAIDAMOUNT	EQ		PAIDNET	PAIDGROSS
	05	A	PUT				RECEIPT	RECEIPT
	06	A	GET				BILLSTUB	BILLSTUB
	07	A	GO TO				TAB0006	TAB0006
	08	A						

Situation 2

return now to the procedure description.

The GET BILLSTUB action, in combination with information supplied by the data description, means to make available the next record in the file named BILLSTUB. This action reads in new values of the variables named PAID, INPUTNAME, PAIDNET, and PAIDGROSS (as would be indicated in the data description). The GET and PUT action operators are exactly the same, and provide the same results, as in Autocoder III since the same IOCS is called into effect.

The last action is another new one, GO TO. This causes a transfer of control to the table identified by the operand. In this case, it is assumed that the current table is table 0006, so this same table is simply repeated with new values of the fields (obtained by the GET). A GO TO always transfers to the beginning of a table.

There is no provision for executing a pre-selected rule of a multiple-rule table. There are, however, provisions for unconditional, i.e., one rule, tables. For example, in Situation 2 the question of how to read the first card is ignored. This would be handled easily enough by an unconditional table, examples of which will be given later.

QUESTIONS

1. List the field names in this table.
2. List the file names.
3. Can it be determined, simply by inspecting this table alone, that INPUTNAME is a field and BILLSTUB is a file?

PAGE	LINE	CC	ACTION OPERATOR	NAME 1	OP	NAME 2	RULE 1	RULE 2
	01	A		PAID	EQ		'N'	'G'
	02	A						
	03	A	SET	CUSTNAME	EQ		INPUT NAME	INPUT NAME
	04	A	SET	PAIDAMOUNT	EQ		PAIDNET	PAIDGROSS
	05	A	PUT				RECEIPT	RECEIPT
	06	A	GET				BILLSTUB	BILLSTUB
	07	A	GO TO				TAB0006	TAB0006
	08	A						

Situation 2

PAGE	LINE	CC	ACTION OPERATOR	NAME 1	OP	NAME 2	RULE 1	RULE 2	RULE 3	RULE 4
	01	A		AGE			LE25	LE25	GR25	GR25
	02	A		SEX	EQ		'M'	'F'		
	03	A		ACCIDENTS					EQ0	GR0
	04	A								
	05	A	SET	RATE			+ RISKFACTOR		- SPCRATFCTR	
	06	A	SET	PREMRATE	EQ		RATE	RATE	RATE	RATE
	07	A	GO TO	COMPCTDISC			X	X	X	X
	08	A								

Situation 3

4. What would happen if the GET and PUT actions were interchanged?

5. What would happen if the last action were GO TO TAB 0941?

Situation 3 involves an automobile insurance liability rate computation, and introduces several new ideas in table structure. First, more than one condition must be satisfied before the actions in a rule can be executed. The condition part of the first rule, for instance, asks, "Is the applicant less than or equal to 25 years of age and a male?" The blank entry in Rule 1 for the accidents condition row means that the condition is not relevant for this rule: a man of 25 or under gets the risk factor, RISKFACTOR, added to his rate regardless of his driving record.

Notice that there is no combination of age, sex, and accident record that satisfies the conditions of more than one rule. This is a requirement of the system: the conditions must be set up so that at most one rule is carried out each time a table is executed.

This table introduces three of the relational operators: EQ, LE, GR. The seven available relational operators and their abbreviations are:

Equal	EQ	Less than (lesser)	LR
Unequal	UN	Greater than or equal to	GE
Greater than	GR	Less than or equal to	LE
Versus	VS		

Answers - Situation 2

1. PAID, CUSTNAME, INPUTNAME, PAIDAMOUNT, PAIDNET, PAIDGROSS.
2. RECEIPT, BILLSTUB.
3. Yes, but only by implication: the operand in a condition must not be a file name, and the operand in a GET or PUT action must be a file name. The data description, if it had been shown, would of course have left no doubt.
4. The order of reading and writing would be interchanged. Actions are always carried out in the sequence in which they appear in a table, from top to bottom.
5. Table 0941, whatever it might be, would be executed next.

LINE	C	ACTION OPERATOR	NAME 1	OP	NAME 2	RULE 1	RULE 2	RULE 3	RULE 4
01	A		AGE			LE25	LE25	GR25	GR25
02	A		SEX	EQ		'M'	'F'		
03	A		ACCIDENTS					EQ0	GR0
04	A								
05	A	SET	RATE			+ RISKFACTOR		- SPCRATFCTR	
06	A	SET	PREMRATE	EQ		RATE	RATE	RATE	RATE
07	A	GO TO	COMPCTDISC			X	X	X	X
08	A								

Situation 3

These operators are to be understood in the sense of a question to which the answer can be only yes or no. The first condition in Rule 1 thus reads: "Is the age of the applicant less than or equal to 25 years?" If the answer is "yes", the condition is satisfied.

The SET operator has a new twist in the first action row: it is used as a two-address arithmetic operator. It is assumed that the RATE has already been computed in a previous table; the table here merely applies special factors if necessary. Rule 1 then says (if the conditions are satisfied):

Set RATE = RATE + RISKFACTOR

The risk factor is added to the rate calculated in a previous table, thus incrementing its value. In Rule 2 there is no second operand for this action, so nothing is done to the rate; likewise in Rule 4. In Rule 3, the action specified is to subtract a special rate factor from the previously computed rate, i. e., decrement the rate by the special risk factor. Note that the EQ is omitted when SET is used to perform arithmetic.

The second action row contains nothing new. The rate, as modified by now if Rule 1 or 3 is being executed, is placed in PREMRATE.

The third action demonstrates the use of a limited entry row in an otherwise extended entry table. In this row, since the same action is to be carried out for each rule, the complete action specification is

PAGE	LINE	C	ACTION OPERATOR	NAME 1	OP	NAME 2	RULE 1 OP	RULE 1 NAME 2	RULE 2 OP	RULE 2 NAME 2	RULE 3 OP	RULE 3 NAME 2	RULE 4 OP	RULE 4 NAME 2
	01	A		AGE			LE	25	LE	25	GR	25	GR	25
	02	A		SEX	EQ			'M'		'F'				
	03	A		ACCIDENTS							EQ	0	GR	0
	04	A												
	05	A	SET	RATE			+	RISKFACTOR			-	SPCRATFCTR		
	06	A	SET	PREMRATE	EQ			RATE		RATE		RATE		RATE
	07	A	GO TO	COMPCTDISC			X		X		X		X	
	08	A												

Situation 3

written in the stub part of the table (to the left of the vertical double line) and X's are placed in the entry part to indicate in which rules that action is to be executed. As explained earlier, this is called limited entry; the format of the other rows is called extended entry, since the statement is "extended" into the entry portion of the table. The second action row could also have been written in limited entry fashion, with SET PREMRATE EQ RATE entirely in the stub, and X's in the "OP" fields of the entries. If an action is not to be executed for a particular rule then the X should be omitted.

The limited entry technique provides no capabilities not available with extended entry, but the simplication of the table is often a convenience.

QUESTIONS

1. State Rules 3 and 4 in English.
2. List the relational operators.
3. In which direction does data flow when the SET... EQ... operator is executed?
4. Does the blank in action 1, Rule 2, mean that RATE is set equal to zero?
5. Could action 2 have been written in limited entry form? What would this have gained?

Answers - Situation 3

1. Rule 3: If the applicant is over 25 and has had no accidents, regardless of sex, decrement the previously computed rate, by the special rate factor, set premium rate to the new rate, and transfer to the compact discount table.

   Rule 4: If the applicant is over 25 and has had one or more accidents, regardless of sex, the previously computed rate is used unchanged. The rate is moved to the premium rate and control is transferred to the table named COMPCTDISC.

2. EQ, UN, GR, LR, GE, LE.

3. Information is moved <u>from</u> the field named as operand 2 <u>to</u> the field named as operand 1, e. g. , SET A EQ 2 means A←2.

4. No. A blank in an action operand field means that no action is taken.

5. Yes, saving a little writing and making it apparent at a glance that the same action is done in each rule.

The preparation of a decision table can be approached in any manner the analyst chooses. Examine the following problem statement:

> If the quantity ordered does not exceed the order limit and if the credit approval is "OK", and if there is a sufficient quantity on hand to fill the order, move the quantity ordered to quantity shipped, and go to a table to prepare ship release.
>
> If the quantity ordered exceeds the order limit, or if the credit approval is not "OK", go to a table named order reject.
>
> If the quantity ordered does not exceed the order limit, and if the credit approval is "OK", but there is insufficient quantity on hand to fill the order, go to a table named back order.

To prepare a decision table for this problem should the condition stub be filled in first, followed by all the actions? Or is the real concern with one condition and one action? Should the table be made in limited or extended entry? Actually there is no set procedure to be followed.

Examine the problem again:

C1

If the Qty Ordered does not exceed the Order Limit/

and

C2

if the Credit Approval is OK/

and

C3

if there is a sufficient Qty On Hand to fill the order/

A1

move the Qty Ordered to Qty Ship/

and

A2

go to Prepare Ship Release/

C4

If the Qty Ordered exceeds the Order Limit/

or

C5

if the Credit Approval is not OK /

A3

go to a table named Order Reject /

C6

If the Qty Ordered does not exceed the Order Limit/

and

C7

if the Credit Approval is OK /

but

C8

there is insufficient Qty On Hand to fill the order/

A4

go to a table named Back Order /

The solid underlines indicate the conditions; the broken underlines indicate actions. Note that the word order is exactly that of the original problem statement. The conditions and actions are numbered here simply as a count; there are eight conditions and four actions. Note, however, that C1, C4 and C6 are concerned with the same operands. Further C2, C5 and C7 involve the same operands. C3 and C8 are also similar.

Spelled out this way, it is obvious enough that there are basically three condition rows and four action rows. Such diagramming is unnecessary and is done here solely for illustrative purposes. The decision table itself is an adequate diagram:

PAGE	LINE	C	ACTION OPERATOR	NAME 1	OP	NAME 2	RULE 1						RULE 2						RULE 3						RULE 4						RULE 5						RULE 6	
							01	02	03	04	05	06	07	08	09	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31	32
	01	A		QTYORDERED	LE	ORDERLIMIT	Y	N	Y	N																												
	02	A		CREDITAPRV	EQ	'OK'	Y		N	Y																												
	03	A		QTYONHAND	GE	QTYORDERED	Y			N																												
	04	A																																				
	05	A	MOVE	QTYORDERED	TO	QTYSHIP	X																															
	06	A	GO TO	ORDEREJECT				X	X																													
	07	A	GO TO	PREPSHPREL			X			X																												
	08	A	GO TO	BACKORDER																																		
	09	A																																				
	10	A																																				
	11	A																																				
	12	A																																				
	13	A																																				
	14	A																																				
	15	A																																				
	16	A																																				
	17	A																																				
	18	A																																				
	19	A																																				
	20	A																																				
	21	A																																				
	22	A																																				
	23	A																																				
	24	A																																				
	25	A																																				

Situation 4, A Limited Entry Table

Name 1	Operator	Name 2
QTYORDERED	LE	ORDERLIMIT
CREDITAPRV	EQ	'OK'
QTYONHAND	GE	QTYORDERED

All the condition stubs are stated above. All the action stubs could then be stated.

	QTYORDERED	LE	ORDERLIMIT
	CREDITAPRV	EQ	'OK'
	QTYONHAND	GE	QTYORDERED
SET	QTYSHIP	EQ	QTYORDERED
GO TO	PREPSHPREL		
GO TO	ORDEREJECT		
GO TO	BACKORDER		

The entry conditions can then be stated with the appropriate "yes" and "no" entries, and finally the action entries are completed by filling in the proper X's for execute. The final result is shown on the left page.

Note that all conditions and actions are stated in limited entry form. When this is the case, the two position columns may be used for the rules, thus allowing many more rules on a single page.

In some situations it may seem more logical as a first step to state a condition, an action and then an entry as below:

Operator	Name 1	OP	Name 2	Rule 1
	CREDITAPRV	EQ	'OK'	N
GOTO	ORDEREJECT			X

IBM 7080

DECISION TABLE LOGIC SHEET

PROGRAM IDENTIFICATION

PAGE	LINE	CC	TYPE	NO. ENTRIES	ELSE	TABLE NO.	IDENTIFICATION	TABLE NAME	NEXT TABLE NAME	ERROR TABLE NAME
16	000	0	M	04	04		TABLE	STKREPORT	GETDATA	

PAGE	LINE	CC		IDENTIFICATION	ORDER OF CONDITION ROWS
	000	1		CONDORDER	

PAGE	LINE	CC		IDENTIFICATION	ORDER OF RULES
16	000	2		RULEORDER	03010204

PROGRAM ____________ DATE ____________

PREPARED BY ____________ PAGE ____________

NEXT PAGE ____________

PAGE	LINE	CC		IDENTIFICATION	COMMENT
16	000	3		TITLE	SET UP AND WRITE REPORT LINE

START NEXT CARD: Dup. Col's. 1–5, Punch B in Col. 6, Skip Col's. 7–33, Dup. Col's. 75–80.

PAGE	LINE	C	ACTION OPERATOR	NAME 1	OP	NAME 2	RULE 1	RULE 2	RULE 3	RULE 4	RULE 5	RULE 6
16	01	A		STOCKTYPE	EQ		01	02	03			
	02	A		SHARES	GR	100000	Y		N			
	03	A										
	04	A	SET	TYPE	EQ		'INDIVID'	'BANK'	'BROKER'			
	05	A	MOVE	STOCKVALUE	TO		VALUE1	VALUE2	VALUE2			
	06	A	PUT	REPORT			X	X	X			
	07	A	GO TO							TAB0006		
	08	A										

Situation 5, Decision Table

There could be other methods of stating the decision table which would be equally valid. The table could be shortened, for example, by combining the three GO TO commands into one extended entry row as below:

Action Operator	Rule 1	Rule 2	Rule 3	Rule 4
GO TO	PREPSHPREL	BACKORDER	ORDEREJECT	ORDEREJECT

In summary, the preparation of a decision table is not a cut and dried procedure, since any approach forces the analyst into a logical analysis of all aspects of the problem.

Situation 5 is the preparation of a stock classification listing.

This table again exhibits multiple conditions, with one of them in limited entry form. The second condition stub asks: "Is the number of shares greater than 100,000?" To satisfy Rule 1, the answer must be yes. In Rule 2, with its blank for this condition, it doesn't matter. In Rule 3, the answer must be no in order to satisfy the condition -- that is, the number of shares must be equal to or less than 100,000. (Note that numeric literals in a table must not contain commas.)

The three rules here do not exhaust all the combinations of the variables in the conditions. For instance, the stock type might not be 1, 2, or 3, or even if the stock type is 1 the number of shares might not be greater than 100,000. In this table provisions for such possibilities

IBM 7080

DECISION TABLE LOGIC SHEET

PROGRAM IDENTIFICATION

PAGE	LINE	CC	TYPE	NO. RULES	ELSE	TABLE NO.	IDENTIFICATION	TABLE NAME	NEXT TABLE NAME	ERROR TABLE NAME
16	000	0	M	04	04		TABLE	STKREPORT	GETDATA	

PAGE	LINE	CC	IDENTIFICATION	ORDER OF CONDITION ROWS
	000	1	CONDORDER	

PAGE	LINE	CC	IDENTIFICATION	ORDER OF RULES
16	000	2	RULEORDER	03010204

PROGRAM ____________ DATE ____________

PREPARED BY ____________ PAGE ____________

NEXT PAGE ____________

PAGE	LINE	CC	IDENTIFICATION	COMMENT
16	000	3	TITLE	SET UP AND WRITE REPORT LINE

START NEXT CARD: Dup. Col's. 1–5, Punch B in Col. 6, Skip Col's. 7–23, Dup. Col's. 75–80.

PAGE	LINE	CC	ACTION OPERATOR	NAME 1	OP	NAME 2	RULE 1 OP	RULE 1 NAME 2	RULE 2 OP	RULE 2 NAME 2	RULE 3 OP	RULE 3 NAME 2	RULE 4 OP	RULE 4 NAME 2	RULE 5 OP	RULE 5 NAME 2	RULE 6 OP	RULE 6 NAME 2
16	01	A		STOCKTYPE	EQ			01		02		03						
	02	A		SHARES	GR	100000	Y				N							
	03	A																
	04	A	SET	TYPE	EQ			'INDIVID'		'BANK'		'BROKER'						
	05	A	MOVE	STOCKVALUE	TO			VALUE1		VALUE2		VALUE2						
	06	A	PUT	REPORT			X		X		X							
	07	A	GO TO											TAB0006				
	08	A																

Situation 5, Decision Table

are made with a special designation called ELSE. This rule says what to do if none of the other rules can be executed. In this example, it is reasonable to expect that it will frequently happen that none of the other rules will be satisfied, so that the ELSE is a more or less normal occurrence. In other tables, failure to satisfy any of the rules might represent an error in the logic of the table or in the data. In such a case no provision would be made for an ELSE; now, if no rule is satisfied, the program proceeds to the error table named in the table header. The table header contains in addition to other descriptive and identification data, a two position field where the rule number for the ELSE is specified. Rule 04 is specified in this case in the table header, and rule 04 states GO TO another table to handle this possibility. In this case, because the ELSE rule takes care of all other possibilities, the entry in the table header for error table name is left blank.

Since it is possible to set up many tables that will be executed in the proper sequence to carry out the necessary processing, there are entries in the table header which must be filled in to relate the various tables. These entries are largely self-explanatory. For example, the column "No. Rules" means just that -- how many rules are there in this table?

Tables are named (or numbered) and these names are used when the table is referenced in the program (i. e., on another decision table

IBM 7080

DECISION TABLE LOGIC SHEET

PROGRAM IDENTIFICATION

PAGE	LINE	CC	TYPE	NO. RULES	ELSE	TABLE NO.	IDENTIFICATION	TABLE NAME	NEXT TABLE NAME	ERROR TABLE NAME
16	000	0	M	04	04		TABLE	STKREPORT	GETDATA	

PAGE	LINE	CC		IDENTIFICATION	ORDER OF CONDITION ROWS
	000	1		CONDORDER	

PAGE	LINE	CC		IDENTIFICATION	ORDER OF RULES
16	000	2		RULEORDER	03010204

PROGRAM ______ DATE ______

PREPARED BY ______ PAGE ______

NEXT PAGE ______

PAGE	LINE	CC		IDENTIFICATION	COMMENT
16	000	3		TITLE	SET UP AND WRITE REPORT LINE

START NEXT CARD: Dup. Col's. 1–5, Punch B in Col. 6, Skip Col's. 7–33, Dup. Col's. 75–80.

PAGE	LINE	C	ACTION OPERATOR	NAME 1	OP	NAME 2	RULE 1	RULE 2	RULE 3	RULE 4	RULE 5	RULE 6
16	01	A		STOCKTYPE	EQ		01	02	03			
	02	A		SHARES	GR	100000	Y		N			
	03	A										
	04	A	SET	TYPE	EQ		'INDIVID'	'BANK'	'BROKER'			
	05	A	MOVE	STOCKVALUE	TO		VALUE1	VALUE2	VALUE2			
	06	A	PUT	REPORT			X	X	X			
	07	A	GO TO							TAB0006		
	08	A										

Situation 5, Decision Table

or within the table itself). Both a name and a number are not necessary, but if the table is numbered, then when the number is referenced within a decision table, the letters TAB must prefix the number, as in the final action of Situation 5.

The letters "$^{C}_{C}$" in the table header stand for "card continuation," which is always zero in the table headers.

Five identification letters can be specified under the column head "Type" which designate:

L -- limited entry

M -- mixed entry

A -- information to follow is in Autocoder

R -- information to follow is report specification

D -- information to follow is data definition.

The details of the table header are given in Chapter 4. Suffice it to say here that the ordering of rules and of condition rows is a programmer's convenience to assure an efficient object program. Finally, the "Comments" section allows the programmer to make any notations he cares to.

To return to the table itself, the first action operator in the procedure table is a familiar one, in this case setting a print field equal to an alphameric literal.

The second action operator is a new one, MOVE . . . TO. This

IBM 7080

DECISION TABLE LOGIC SHEET

PROGRAM IDENTIFICATION

PAGE	LINE	CC	TYPE	NO. RULES	ELSE	TABLE NO.	IDENTIFICATION	TABLE NAME	NEXT TABLE NAME	ERROR TABLE NAME
16	000	0	N	04	04		TABLE	STKREPORT	GETDATA	

PAGE	LINE	CC			IDENTIFICATION	ORDER OF CONDITION ROWS
	0001				CONDORDER	

PAGE	LINE	CC			IDENTIFICATION	ORDER OF RULES
16	0002				RULEORDER	03010204

PAGE	LINE	CC		IDENTIFICATION	COMMENT
16	0003			TITLE	SET UP AND WRITE REPORT LINE

PROGRAM ______ DATE ______

PREPARED BY ______ PAGE ______

NEXT PAGE ______

START NEXT CARD: Dup. Col's. 1–5, Punch B in Col. 6, Skip Col's. 7–33, Dup. Col's. 75–80.

PAGE	LINE	C	ACTION OPERATOR	NAME 1	OP	NAME 2	RULE 1	RULE 2	RULE 3	RULE 4	RULE 5	RULE 6
16	01	A		STOCKTYPE	EQ		01	02	03			
	02	A		SHARES	GR	100200	Y		N			
	03	A										
	04	A	SET	TYPE	EQ		'INDIVID'	'BANK'	'BROKER'			
	05	A	MOVE	STOCKVALUE	TO		VALUE1	VALUE2	VALUE2			
	06	A	PUT	REPORT			X	X	X			
	07	A	GO TO							TAB0006		
	08	A										

Situation 5, Decision Table

IBM 7080

DECISION TABLE NAME DESCRIPTION SHEET

PROGRAM ______

DATE ______

PREPARED BY ______

PAGE ______

NEXT PAGE ______

PG	LINE	CC	TYP		TABLE NO.	IDENT	TABLE NAME	
	0,0,0	0				TABLE		

IDENT.

PG	LINE	NAME	CLASS	NUM.	DESCRIPTION	COMMENTS
	01					
	02					
	03					
	04	STOCKTYPE	FLD	02	N	
	05	SHARES		08	N	
	06	STOCKVALUE		10	#+08.02	
	07					
	08					
	09	TYPE	FLD	07	A+	
	10	VALUE1	RPT	15	$XX,XXX,XXZ.ZZ	¤$¤
	11	VALUE2	RPT	14	$XX,XXX,XXZ.ZZ	¤*¤CR¤
	12					

Situation 5, Name Description

is the opposite of SET . . . EQ: it moves the quantity specified by the first operand to the field named in the second operand. The difference in usefulness of the two operators may be stated as follows: SET . . . EQ is valuable when one of several quantities in the second operand field is to be moved to the first, common, operand field; MOVE . . . TO is valuable when a quantity in the first operand field is to be transferred to one of several fields named by subsequent operands.

The third operator is the now familiar terminology of the 7080 Decision Table System which means to make data available for the writing of an output report. This action is expressed in limited entry form.

The names for the variables in this table introduce some additional processing that is not evident from a study of the decision logic sheet. As previously mentioned, Decision Table Name Description Sheets are a convenience so that the decision table can deal with names. The description forms are required for such details as classifying data, specifying its length and special characteristics, and specifying values.

Consider the Decision Table Name Description Sheet shown for Situation 5. The letters "FLD" in the "Class" column indicate that this name is a field as opposed to a file, group name, report, etc.

IBM 7080 — DECISION TABLE LOGIC SHEET

PAGE	LINE	TABLE NO.	IDENTIFICATION	TABLE NAME	NEXT TABLE NAME	ERROR TABLE NAME
16	0000	M0404	TABLE	STKREPORT	GETDATA	

PAGE	LINE	IDENTIFICATION	ORDER OF CONDITION ROWS
	0001	CONDORDER	

PAGE	LINE	IDENTIFICATION	ORDER OF RULES
16	0002	RULEORDER	03010204

PAGE	LINE	IDENTIFICATION	COMMENT
16	0003	TITLE	SET UP AND WRITE REPORT LINE

PROGRAM ______ DATE ______
PREPARED BY ______ PAGE ______
NEXT PAGE ______

START NEXT CARD: Dup. Col's. 1–5, Punch B in Col. 6, Skip Col's. 7–33, Dup. Col's. 75–80.

PAGE	LINE	C	ACTION OPERATOR	NAME 1	OP	NAME 2	RULE 1	RULE 2	RULE 3	RULE 4	RULE 5	RULE 6
16	01	A		STOCKTYPE	EQ		01	02	03			
	02	A		SHARES	GR	100000	Y		N			
	03	A										
	04	A	SET	TYPE	EQ		'INDIVID'	'BANK'	'BROKER'			
	05	A	MOVE	STOCKVALUE	TO		VALUE1	VALUE2	VALUE2			
	06	A	PUT	REPORT			X	X	X			
	07	A	GO TO							TAB0006		
	08	A										

Situation 5, Decision Table

IBM 7080 — DECISION TABLE NAME DESCRIPTION SHEET

PROGRAM ______
DATE ______
PREPARED BY ______
PAGE ______
NEXT PAGE ______

PG	LINE	TABLE NO.	IDENT	TABLE NAME
	0,0,0		TABLE	

PG	LINE	NAME	CLASS	NUM.	DESCRIPTION	COMMENTS
		STOCKTYPE	FLD	02	N	
		SHARES		08	N	
		STOCKVALUE		10	#+08.02	
		TYPE	FLD	07	A+	
		VALUE1	RPT	15	$XX,XXX,XXZ.ZZ ¤$¤	
		VALUE2	RPT	14	$XX,XXX,XXZ.ZZ ¤*¤CR¤	

Situation 5, Name Description

The number "2" in the number column indicates that it is 2 digits in length and the letter "N" under "Description" states that it is numeric.

The name SHARES is classified as a field since the "class" column is blank; the assumption is made that it is in the same class as the preceding entry. It is 8 digits in length and numeric.

Under the description of STOCK VALUE, the first position contains a number sign (#). This signals the processor that a decimal point is to be defined for a quantity of a specified number with integers and decimal positions. In this case a signed number with two decimal digits and eight integers is specified. Alternatively this ten digit field could have been represented with the more conventional X's with the decimal point located in its proper position (+XXXXXXXX.XX). TYPE is also classified as a field. The code characters A+ define this field as a non-numerical field whose right-most character will always provide left protection for an adjacent signed field. In the absence of left protection, the single character A would be specified.

VALUE1 and VALUE2 are descriptions for variables in a printed report. Provisions are made for the suppression of insignificant zeros in the report by the placement of Z's in non-affected positions. Commas are used in the normal manner as they must always be; i. e., the fourth position to the left of a decimal point or sign indicator and then after

IBM 7080 **DECISION TABLE LOGIC SHEET**

PROGRAM IDENTIFICATION

PAGE	LINE	CC	TYPE	NO. RULES	ELSE	TABLE NO.	IDENTIFICATION	TABLE NAME	NEXT TABLE NAME	ERROR TABLE NAME
16	000	0	M	04	04		TABLE	STKREPORT	GETDATA	

PAGE	LINE	CC	IDENTIFICATION	ORDER OF CONDITION ROWS
	000	1	CONDORDER	

PAGE	LINE	CC	IDENTIFICATION	ORDER OF RULES
16	000	2	RULEORDER	03010204

PROGRAM ______ DATE ______

PREPARED BY ______ PAGE ______

PAGE	LINE	CC	IDENTIFICATION	COMMENT
16	000	3	TITLE	SET UP AND WRITE REPORT LINE

NEXT PAGE ______

START NEXT CARD: Dup. Col's. 1–5, Punch B in Col. 6, Skip Col's. 7–33, Dup. Col's. 75–80.

PAGE	LINE	C	ACTION OPERATOR	NAME 1	OP	NAME 2	RULE 1	RULE 2	RULE 3	RULE 4	RULE 5	RULE 6
16	01	A		STOCKTYPE	EQ		01	02	03			
	02	A		SHARES	GR	100000	Y		N			
	03	A										
	04	A	SET	TYPE	EQ		'INDIVID'	'BANK'	'BROKER'			
	05	A	MOVE	STOCKVALUE	TO		VALUE1	VALUE2	VALUE2			
	06	A	PUT	REPORT			X	X	X			
	07	A	GO TO							TAB0006		
	08	A										

Situation 5, Decision Table

IBM 7080 **DECISION TABLE NAME DESCRIPTION SHEET**

PROGRAM ______

DATE ______

PREPARED BY ______

PAGE ______

NEXT PAGE ______

PG	LINE	CC	TYP	TABLE NO.	IDENT	TABLE NAME
	0000	0			TABLE	

IDENT.

PG	LINE	NAME	CLASS	NUM.	DESCRIPTION	COMMENTS
		STOCKTYPE	FLD	02	N	
		SHARES		08	N	
		STOCKVALUE		10	#+08.02	
		TYPE	FLD	07	A+	
		VALUE1	RPT	15	$XX, XXX, XXZ. ZZ ¤$¤	
		VALUE2	RPT	14	$XX, XXX, XXZ. ZZ	¤*¤CR¤

Situation 5, Name Description

every three additional positions to the left. The **15** in the number column includes a count of all the positions in the operand up to the first lozenge (⌑); i. e., the dollar sign, eight integers, two decimal places, three commas and one position for sign indication. No sign indication is specified, and the system will automatically provide for a - for minus indication and no sign indication for plus or zero.

VALUE1 demonstrates the use of a floating dollar sign; i. e., one which will appear immediately to the left of the high-order significant digit. One position is allocated for sign indication following the low-order digit and then a floating dollar sign is specified by the character $ written between two lozenges.

VALUE2 introduces two new definitions: CR or credit symbol print-out and asterisk protection. In this print-out the dollar sign always appears in the position assigned regardless of the number of digits printed because a floating dollar sign is not specified. Two digit positions following the low-order integer are left blank for the print-out of the letters CR. The CR enclosed between lozenges indicates that the letters CR are to be printed when the value is negative in the two positions following the right-most digit -- this will be blank if the value is positive.

The asterisk enclosed in lozenges specifies that asterisks are to be printed in all blank positions between the fixed dollar sign and the

IBM 7080

DECISION TABLE LOGIC SHEET

PROGRAM IDENTIFICATION

PAGE	LINE	CC	TYPE	NO. RULES	ELSE	TABLE NO.	IDENTIFICATION	TABLE NAME	NEXT TABLE NAME	ERROR TABLE NAME
16	000	0	M	04	04		TABLE	STKREPORT	GETDATA	

PAGE	LINE	CC			IDENTIFICATION	ORDER OF CONDITION ROWS
	000	1			CONDORDER	

PAGE	LINE	CC			IDENTIFICATION	ORDER OF RULES
16	000	2		RULE	ORDER	03010204

PROGRAM ______ DATE ______

PREPARED BY ______ PAGE ______

PAGE	LINE	CC		IDENTIFICATION	COMMENT
16	000	3		TITLE	SET UP AND WRITE REPORT LINE

NEXT PAGE ______

START NEXT CARD: Dup. Col's. 1–5, Punch B in Col. 6, Skip Col's. 7–33, Dup. Col's. 75–80.

PAGE	LINE	C	ACTION OPERATOR	NAME 1	OP	NAME 2	RULE 1	RULE 2	RULE 3	RULE 4	RULE 5	RULE 6
16	01	A		STOCKTYPE	EQ		01	02	03			
	02	A		SHARES	GR	100000	Y		N			
	03	A										
	04	A	SET	TYPE	EQ		'INDIVID'	'BANK'	'BROKER'			
	05	A	MOVE	STOCKVALUE	TO		VALUE1	VALUE2	VALUE2			
	06	A	PUT	REPORT			X	X	X			
	07	A	GO TO							TAB0006		
	08	A										

Situation 5, Decision Table

IBM 7080

DECISION TABLE NAME DESCRIPTION SHEET

PG	LINE	CC	TYP		TABLE NO.	IDENT	TABLE NAME	
	0000	0				TABLE		

IDENT.

PROGRAM ______

DATE ______

PREPARED BY ______

PAGE ______

NEXT PAGE ______

PG	LINE	NAME	CLASS	NUM.	DESCRIPTION	COMMENTS
		STOCKTYPE	FLD	02	N	
		SHARES		08	N	
		STOCKVALUE		10	#+08.02	
		TYPE	FLD	07	A+	
		VALUE1	RPT	15	$XX, XXX, XXZ. ZZ	¤$¤
		VALUE2	RPT	14	$XX, XXX, XXZ. ZZ	¤*¤CR¤

Situation 5, Name Description

high-order digit -- which again may vary in position.

The editing functions discussed all appear on the Decision Table Name Description Sheet. Observe that there is nothing on the Decision Table Logic Sheet to indicate that editing is to be done.

In summary, Situation 5 has been largely concerned with the necessity for a table header and for a Decision Table Name Description Sheet and has illustrated a few of their functions. More details of their many functions will be given in situations that follow and a complete explanation is given in Chapter 4.

QUESTIONS

1. What is the function of an ELSE condition and under what circumstances is it employed?
2. What is the general purpose of a table header?
3. What is the difference between a SET and MOVE operator?
4. How are editing functions specified using the 7080 DTS?

Answers - Situation 5

1. An ELSE condition provides for an unconditional rule. If none of the other rule conditions are satisfied the ELSE or "all others" rule is executed.

2. The general purpose of a table header is to identify the table and relate it to other tables in the procedure. It also supplies additional information to the processor to aid in compiling the program. Other notations permit the analyst to specify re-ordering for increased object program efficiency without rewriting the table.

3. SET transfers the value of the second operand to the first: MOVE transfers the first to the second, e.g., MOVE A TO B means $A \longrightarrow B$ where SET A EQ B means $A \longleftarrow B$.

4. Editing functions are specified on a form separate from the decision table. This form, the Decision Table Name Description sheet, is used to define operands so that the table itself can deal with names.

IBM 7080 — DECISION TABLE LOGIC SHEET

PROGRAM IDENTIFICATION

PAGE	LINE	CC	TYPE	NO. RULES	ELSE	TABLE NO.	IDENTIFICATION	TABLE NAME	NEXT TABLE NAME	ERROR TABLE NAME
26	000	0	M	04			TABLE	BOND STOCK		TABSTOP

PAGE	LINE	CC		IDENTIFICATION	ORDER OF CONDITION ROWS
	000	1		CONDORDER	

PAGE	LINE	CC		IDENTIFICATION	ORDER OF RULES
	000	2		RULEORDER	

PAGE	LINE	CC		IDENTIFICATION	COMMENT
	000	3		TITLE	

PROGRAM ____ DATE ____

PREPARED BY ____ PAGE ____

NEXT PAGE ____

START NEXT CARD: Dup. Col's. 1–5, Punch B in Col. 6, Skip Col's. 7–33, Dup. Col's. 75–80.

PAGE	LINE	CC	ACTION OPERATOR	NAME 1	OP	NAME 2	RULE 1	RULE 2	RULE 3	RULE 4	RULE 5	RULE 6
26	01	A		PURCHCODE	EQ		DEDLIST	'B'	'S'	'A'		
	02	A										
	03	A	MOVE	EMPLOYINFO	TO	EMPLY IDENT		X	X	X		
	04	A	SET	STKBALNCE	+	STKAMOUNT			X			
	05	A	DO					BUYBONDS	BUYSTOCK	BUYBONDS		
	06	A	DO							BUYSTOCK		
	07	A	GO TO				OTHERDED	TAB0260	TAB0260	TAB0260		
	08	A										

Situation 6, Open Decision Table

Situation 6 is one aspect of a payroll operation. The table header states that there are four rules and that it is a mixed entry decision table. No ELSE rule is specified as all normal conditions have been covered, and failure to satisfy any of the rules would be an error. If a special error routine had been set up this would have been specified under "Error Table Name" in the table header. In this decision table the special operand TABSTOP is specified. TABSTOP causes a program halt whether specified under "Error Table Name" or "Next Table Name" in the header, or in a GO TO statement as an operand in the table itself.

The single condition row compares the purchase code with three alphameric literals (which do not require description) in rules 2, 3, and 4. In rule 1, however, the extended entry condition operand DEDLIST requires definition, since it is a name designated by the analyst. The definition is not for a variable field, but for a fixed set of values. Fixed values are not described on the same Name Description sheet as variable information. Description Table 0003 is used to define fixed values. In the case of DEDLIST the reference is to a list of constant values used only in the condition section of the decision table. The class entry VLIST (Value List) is always used for such a list with the values listed as shown in the example.

Rule 1 states that if the value of PURCHCODE in the current record is any one of the entries listed on the value list, then GO TO a table named

IBM 7080 **DECISION TABLE LOGIC SHEET** PROGRAM IDENTIFICATION

PAGE	LINE	CC	TYPE	NO. RULES	ELSE	TABLE NO.	IDENTIFICATION	TABLE NAME	NEXT TABLE NAME	ERROR TABLE NAME
26	000	0	M	04			TABLE	BOND STOCK	TABSTOP	

PAGE	LINE	CC	IDENTIFICATION	ORDER OF CONDITION ROWS
	000	1	CONDORDER	

PAGE	LINE	CC	IDENTIFICATION	ORDER OF RULES
	000	2	RULEORDER	

PROGRAM ____________ DATE ____________

PREPARED BY ____________ PAGE ____________

PAGE	LINE	CC	IDENTIFICATION	COMMENT
	000	3	TITLE	

NEXT PAGE ____________

START NEXT CARD: Dup. Col's. 1–5, Punch B in Col. 6, Skip Col's. 7–33, Dup. Col's. 75–80.

PAGE	LINE	CC	ACTION OPERATOR	NAME 1	OP	NAME 2	RULE 1 OP	RULE 1 NAME 2	RULE 2 OP	RULE 2 NAME 2	RULE 3 OP	RULE 3 NAME 2	RULE 4 OP	RULE 4 NAME 2	RULE 5 OP	RULE 5 NAME 2	RULE 6 OP	RULE 6 NAME 2
26	01	A		PURCHCODE	EQ			DEDLIST		'B'		'S'		'A'				
	02	A																
	03	A	MOVE	EMPLOYINFO	TO	EMPLY.IDENT			X		X		X					
	04	A	SET	STKBALNCE	+	STKAMOUNT					X							
	05	A	DO							BUYBONDS		BUYSTOCK		BUYBONDS				
	06	A	DO											BUYSTOCK				
	07	A	GO TO					OTHERDED		TAB0260		TAB0260		TAB0260				
	08	A																

Situation 6, Open Decision Table

IBM 7080 **DECISION TABLE NAME DESCRIPTION SHEET**

PG	LINE	CC	TYP	TABLE NO.	IDENT	TABLE NAME
10	0000	0	D	0003	TABLE	MISC

PROGRAM ____________

DATE ____________

PREPARED BY ____________

PAGE ____________

NEXT PAGE ____________

IDENT. 75 80

PG	LINE	NAME	CLASS	NUM.	DESCRIPTION	COMMENTS
10	01	DEDLIST	VLIST	1	I	INSURANCE
	02				L	LOANS
	03				C	CREDIT UNION
	04				D	DUES
	05				N	NONE
	06					
	07	EMPLOYINFO	SEXP	32	JOIN(LOCATION, DEPT, EMPLOYEENO, NAME)	
	08	MARKTVALUE	MIXP		#04.02, (DAYSHIGH + DAYSLOW) / 2	

Situation 6, Table 0003

PG	LINE	NAME	CLASS	NUM.	DESCRIPTION	COMMENTS
	01	PURCHASE	FILE		TAPE 2008	1 RECORD/BLOCK
	02		RNAME		APURCHRM	50 CHAR, FIXED LENTH
	03	EMPLY.IDENT	GNAME		NAMEP	LOCATION, DEPT, EMPLOYEE NO, NAME
	04	LOCATIONP	FLD	3	N	
	05	DEPTP		3	N	
	06	EMPLOYNOP		6	N	
	07	NAMEP		20	A	
	08	PURCHPRICE		6	#+04.02	PURCHASE PRICE-BOND or STOCK
	09	SBALNCE		6	#+04.02	NEW STOCK BALANCE
	10	PURCH.CODE		1	A	PURCHASE CODE
	11	PURCHRM	PRE	5	‡	ENDING RECORD MARK
	12					

Situation 6, Purchase file

OTHERDED. It it is not on the list, the test will fail.

The first action of rule 2 states MOVE EMPLOYINFO TO EMPLYIDENT. The operator MOVE . . . TO has been previously explained, but two new operand types are involved. EMPLOYINFO is the name of a symbolic expression. Symbolic expressions operate on symbols or strings of characters in much the same way as arithmetic expressions operate on numbers. The symbolic expression calls for the joining together of various data elements and treating them as a single field. By defining the "stringing together" of operands in table 0003, reference is made with one call in this example to LOCATION, DEPARTMENT . . . NAME, without complicating the decision table with calls to each of these fields. Note that the description includes the word JOIN. This 7080 decision table operator is always used in connection with the other system word in the class entry SEXP, and causes all the data elements to be treated as a unit. Variables, constants, literals and other symbolic expressions can be joined end-to-end by the expression, they need not be adjacent initially.

The other new operand type, EMPLYIDENT, is a group name -- classified as a GNAME. GNAMES, like symbolic expressions allow multiple operands to be treated as a unit. The distinction between group names and symbolic expressions is two fold: (1) group names deal with continuous sections in memory; (2) symbolic expressions allow strings of characters from discontinuous points in memory to be brought together

IBM 7080 **DECISION TABLE LOGIC SHEET**

PROGRAM IDENTIFICATION

PAGE	LINE	CC	TABLE NO.	IDENTIFICATION	TABLE NAME	NEXT TABLE NAME	ERROR TABLE NAME
26	0000	M	04	TABLE	BOND STOCK		TABSTOP

PAGE	LINE	CC	IDENTIFICATION	ORDER OF CONDITION ROWS
	0001		CONDORDER	

PAGE	LINE	CC	IDENTIFICATION	ORDER OF RULES
	0002		RULEORDER	

PAGE	LINE	CC	IDENTIFICATION	COMMENT
	0003		TITLE	

PROGRAM ____ DATE ____

PREPARED BY ____ PAGE ____

NEXT PAGE ____

START NEXT CARD: Dup. Col's. 1–5, Punch 8 in Col. 6, Skip Col's. 7–33, Dup. Col's. 75–80.

PAGE	LINE	C	ACTION OPERATOR	NAME 1	OP	NAME 2	RULE 1	RULE 2	RULE 3	RULE 4	RULE 5	RULE 6
26	01	A		PURCHCODE	EQ		DEDLIST	'B'	'S'	'A'		
	02	A										
	03	A	MOVE	ENPLOYINFO	TO	EMPLY IDENT		X	X	X		
	04	A	SET	STKBALNCE	+	STKAMOUNT			X			
	05	A	DO					BUYBONDS	BUYSTOCK	BUYBONDS		
	06	A	DO							BUYSTOCK		
	07	A	GO TO				OTHERDED	TAB0260	TAB0260	TAB0260		
	08	A										

Situation 6, Open Decision Table

PG	LINE	NAME	CLASS	NUM.	DESCRIPTION	COMMENTS
05	01	DEDUCTION	FILE		TAPE-2006, 2007	20 RECORD/BLOCK
	02		RNAME		ADEDRM	60 CHAR, FIXED LENTH
	03	NAME	FLD	20A		
	04	EMPLOYEENO		6N		
	05	BONDAMOUNT		5#+03.02		
	06	STKBALNCE		6#+04.02		
	07	STKAMOUNT		5#+03.02		
	08	PURCHCODE		1A		
	09	DEPT		3N		
	10	LOCATION		5N		
	11	DEDRM	PRE	1#		
	12					

Situation 6, Deduction file

IBM 7080 DECISION TABLE NAME DESCRIPTION SHEET

PROGRAM ____

DATE ____

PREPARED BY ____

PAGE ____

NEXT PAGE ____

PG	LINE	CC	TYP	TABLE NO.	IDENT	TABLE NAME
27	0000	0	A		TABLE	BUYBONDS

PG	LINE	NAME	CLASS	NUM.	DESCRIPTION	COMMENTS
	01		LOD	01	#B#	
	02		UNL	01	PURCHCODE	'B' → PURCHASE CODE
	03		RAD		BONDAMOUNT	EMPLOYEE MUST DEDUCT EXACT BOND
	04		SET		6	AMOUNT - - 18.75, 37.50, 75.00
	05		STO		PURCHPRICE	BONDAMOUNT → PURCHASE CODE 00XX.XX
	06		BLANK		SBALNCE	BLANKS → STOCK BALANCE
	07		PUT		PURCHASE	WRITE PURCHASE ON CARD IMAGE TAPE
	08					

Situation 6, Closed Autocoder Table

and treated as a unit. The sequence of characters is in the order in which the operands are specified in the symbolic expression itself.

The first action row of rule 2, then, moves the current values of LOCATION, T, NAME . . . to a 32 position area called EMPLY-IDENT.

The second action row introduces a new action operator -- DO. DO is a sequence control operator which causes the execution of the table named as the operand, and upon execution of this "closed table", control is returned to the table which referenced the closed table. In contrast an "open table" takes control without reverting to the table which referenced it. Open tables are never called with a DO command.

The closed table referenced by action row 2 of rule 2 is named BUYBONDS. This is a pseudo table as it is written in Autocoder language on an operand description sheet. It references data described on other operand description tables. Such procedures must be closed routines. Note that in the table header this sheet has as the "Type" an entry of the letter A. This indicates to the processor that this is an Autocoder pseudo table. The letter "D" would be the type code for description sheets which describe operands

The final action of rule 2 in situation 6 is the familiar GO TO operator and the operand is TAB0260.

In rule 3 the condition is the PURCHCODE equals the literal 'S'. The first action row is identical to the operation described for rule 2; i. e.,

IBM 7080 **DECISION TABLE LOGIC SHEET**

PROGRAM IDENTIFICATION

PAGE	LINE	C	TABLE NO.	IDENTIFICATION	TABLE NAME	NEXT TABLE NAME	ERROR TABLE NAME
26	000	0	M04	TABLE	BOND STOCK		TABSTOP

PAGE	LINE	C	IDENTIFICATION	ORDER OF CONDITION ROWS
	000	1	CONDORDER	
	000	2	RULEORDER	ORDER OF RULES
	000	3	TITLE	COMMENT

PROGRAM ________ DATE ________

PREPARED BY ________ PAGE ________

NEXT PAGE ________

START NEXT CARD: Dup. Col's. 1–5, Punch B in Col. 6, Skip Col's. 7–33, Dup. Col's. 75–80.

PAGE	LINE	C	ACTION OPERATOR	NAME 1	OP	NAME 2	RULE 1	RULE 2	RULE 3	RULE 4	RULE 5	RULE 6
26	01	A		PURCHCODE	EQ		DEDLIST	'B'	'S'	'A'		
	02	A										
	03	A	MOVE	EMPLOYINFO	TO	EMPLY IDENT		X	X	X		
	04	A	SET	STKBALNCE	+	STKAMOUNT			X			
	05	A	DO					BUYBONDS	BUYSTOCK	BUYBONDS		
	06	A	DO							BUYSTOCK		
	07	A	GO TO				OTHERDED	TAB0260	TAB0260	TAB0260		
	08	A										

Situation 6, Open Decision Table

PG	LINE	NAME	CLASS	NUM.	DESCRIPTION	COMMENTS
05	01	DEDUCTION	FILE		TAPE-2006,2007	20 RECORD/BLOCK
	02		RNAME		ADEDRM	60 CHAR, FIXED LENTH
	03	NAME	FLD	20A		
	04	EMPLOYEE NO		6N		
	05	BONDAMOUNT		5#+03.02		
	06	STKBALNCE		6#+04.02		
	07	STKAMOUNT		5#+03.02		
	08	PURCHCODE		1A		
	09	DEPT		3N		
	10	LOCATION		3N		
	11	DEDRM	PRE	1#		
	12					

Situation 6, Deduction file

PAGE	LINE	C	TABLE NO.	IDENTIFICATION	TABLE NAME	NEXT TABLE NAME	ERROR TABLE NAME
28	000	0	L02	TABLE	BUYSTOCK	DO	TAB TOP

PAGE	LINE	C	ACTION OPERATOR	NAME 1	OP	NAME 2	RULE 1	RULE 2	RULE 3	RULE 4	RULE 5	RULE 6
28	01	A		STKBALNCE	GE	MARKTVALUE	Y N					
	02	A										
	03	A	SET	PURCHPRICE	EQ	MARKTVALUE	X					
	04	A	SET	STKBALNCE	-	PURCHPRICE	X					
	05	A	MOVE	'S'	TO	PURCHCODE	X					
	06	A	MOVE	STKBALNCE	TO	SBALNCE	X					
	07	A	PUT	PURCHASE			X					
	08	A										

Situation 6, Closed Decision Table

transferring a symbolic expression value to a group name of the PURCHASE file.

The second action specified in rule 3 is a two-address arithmetic command operating upon two data fields in the input/output area: STK-BALNCE and STKAMOUNT. These unassigned variables have names assigned to them; their length, format and their special characteristics (i.e., sign, decimal position, etc.) are defined in the operand description sheet for the input/output areas.

The third action specified in rule 3, as in rule 2, calls for the execution of a closed decision table, so that control will be returned to the calling table. Note that the BUYSTOCK table has in its table header under "Next Table Name" the entry DO -- specifying that it is a closed decision table.

Consider now the closed decision table BUYSTOCK. This limited entry table has one condition row and two rules. The first operand of the condition is STKBALNCE. This field, classified as FLD, and described in the input/output operand description is part of the file named DEDUCTION. This entire file can be referenced by referring to its name with input/output actions, or as in this condition statement a single defined field is referenced. Note also that all of the fields in the file are classified with the entry RNAME meaning record name. If there had been occasion in this program to reference the record as an entity, e.g. with a MOVE, a name would have been given the RNAME entry.

PAGE	LINE	CC	TYPE	NO. RULES	ELSE	TABLE NO.	IDENTIFICATION	TABLE NAME	NEXT TABLE NAME	ERROR TABLE NAME
28	000	0	L	02			TABLE	BUYSTOCK	DO	TAB_TOP

PAGE	LINE	C	ACTION OPERATOR	NAME 1	OP	NAME 2	RULE 1 01	02	03	04	05	06	RULE 2 07	08	09	10	11	12	RULE 3	RULE 4	RULE 5	RULE 6
28	01	A		STKBALNCE	GE	MARKTVALUE	Y	N														
	02	A																				
	03	A	SET	PURCHPRICE	EQ	MARKTVALUE	X															
	04	A	SET	STKBALNCE	-	PURCHPRICE	X															
	05	A	MOVE	'S'	TO	PURCHCODE	X															
	06	A	MOVE	STKBALNCE	TO	SBALNCE	X															
	07	A	PUT	PURCHASE			X															
	08	A																				

Situation 6, Closed Decision Table

IBM 7080 DECISION TABLE NAME DESCRIPTION SHEET

PROGRAM ____
DATE ____
PREPARED BY ____
PAGE ____
NEXT PAGE ____

PG	LINE	CC	TYP	TABLE NO.	IDENT	TABLE NAME
10	000	0	D	0003	TABLE	MISC

IDENT.

PG	LINE	NAME	CLASS	NUM.	DESCRIPTION	COMMENTS
10	01	DEDLIST	VLIST	1	I	INSURANCE
	02				L	LOANS
	03				C	CREDIT UNION
	04				D	DUES
	05				N	NONE
	06					
	07	EMPLOYINFO	SEXP	22	JOIN(LOCATION, DEPT, EMPLOYEENO, NAME)	
	08	MARKTVALUE	MIXP		#04.02, (DAYSHIGH + DAYSLOW) / 2	

Situation 6, Table 0003

PG	LINE	NAME	CLASS	NUM.	DESCRIPTION	COMMENTS
	01	PURCHASE	FILE		TAPE 2008	1 RECORD/BLOCK
	02		RNAME		APURCHRM	60 CHAR, FIXED LENTH
	03	EMPLY.IDENT	GNAME		NAMEP	LOCATION, DEPT, EMPLOYEE NO, NAME
	04	LOCATIONP	FLD	3	N	
	05	DEPTP		3	N	
	06	EMPLOYNOP		6	N	
	07	NAMEP		20	A	
	08	PURCHPRICE		6	#+04.02	PURCHASE PRICE-BOND or STOCK
	09	SBALNCE		6	#+04.02	NEW STOCK BALANCE
	10	PURCH.CODE		1	A	PURCHASE CODE
	11	PURCHRM	PRE	5	‡	ENDING RECORD MARK
	12					

Situation 6, Purchase file

The second operand in the condition statement, MARKTVALUE, is a new type of expression. It is classified in operand description sheet 0003 as MEXP which designates an arithmetic expression. MEXP as an operand designates that the evaluation of the expression is to take place at the time it is used: in this case computing the stock's market value by dividing the sum of the day's high and low selling price by two.

The first four action rows of the decision table named BUYSTOCK involve the now familiar operators SET and MOVE, a two address arithmetic operator, and as operands FLD, MEXP, and a literal.

The fifth action row utilizes the operator: PUT. The operand here, PURCHASE, is a file name, described in the input/output operand description. The PUT operator, with a file name as an operand, causes the record to be released from the program and placed in the output area to be written out automatically by IOCS when a block has been filled.

In summary, Situation 6 is concerned with the relationship of the various tables which are employed in one data processing operation and with the data descriptions associated with these tables.

An open table has called upon a closed table with a DO command, and another DO command has called on an Autocoder pseudo table used as an auxiliary mode accepted by the system and treated as a closed table.

Data description have enabled the tables to employ a single operand to call upon continuous sections in the memory (GNAME) and upon discontinuous fields in memory with a single symbolic expression in

PAGE	LINE	CC	TYPE	NO. RULES	ELSE	TABLE NO.	IDENTIFICATION	TABLE NAME	NEXT TABLE NAME	ERROR TABLE NAME
28	000	0	L	02			TABLE	BUYSTOCK	DO	TAB.TOP

PAGE	LINE	C	ACTION OPERATOR	NAME 1	OP	NAME 2	RULE 1 01	RULE 1 02	RULE 2	RULE 3	RULE 4	RULE 5	RULE 6
28	01	A		STKBALNCE	GE	MARKTVALUE	Y	N					
	02	A											
	03	A	SET	PURCHPRICE	EQ	MARKTVALUE	X						
	04	A	SET	STKBALNCE	-	PURCHPRICE	X						
	05	A	MOVE	'S'	TO	PURCHCODE	X						
	06	A	MOVE	STKBALNCE	TO	SBALNCE	X						
	07	A	PUT	PURCHASE			X						
	08	A											

Situation 6, Closed Decision Table

IBM 7080 DECISION TABLE NAME DESCRIPTION SHEET

PROGRAM ____
DATE ____
PREPARED BY ____
PAGE ____
NEXT PAGE ____

PG	LINE	CC	TYP	TABLE NO.	IDENT	TABLE NAME
10	000	0	D	0003	TABLE	MISC

PG	LINE	NAME	CLASS	NUM.	DESCRIPTION	COMMENTS
10	01	DEDLIST	VLIST	1	I	INSURANCE
	02				L	LOANS
	03				C	CREDIT UNION
	04				D	DUES
	05				N	NONE
	06					
	07	EMPLOYINFO	SEXP	22	JOIN(LOCATION, DEPT, EMPLOYEENO, NAME)	
	08	MARKTVALUE	MIXP		*04.02, (DAYSHIGH + DAYSLOW) / 2	

Situation 6, Table 0003

connection with a JOIN operator(SEXP). Arithmetic expressions (MEXP) have permitted the table to employ a single operand to designate that the computation described by the expression is to be performed.

Situation 6 has also shown that an entire file, a record within the file or a single field can be referenced by its name. A Value List (VLIST) has been used to determine if an input value is the same as a value in the list. The condition is satisfied if there is a corresponding argument on the list.

QUESTIONS

1. What class entry is used in describing a list of constant values and where may such a list be referenced in a decision table?
2. What is a SEXP operand and how does it differ in function from most named operands?
3. When is a GNAME used?
4. What is the only action operator which can reference an Autocoder pseudo table? Why is this restriction made?
5. Where is control resumed after the execution of a closed decision table?

Answers - Situation 6

1. V LIST defines a list of constant values and is used only in the condition section of a decision table.

2. A symbolic expression differs from most named operands in that it causes operations to be performed on the data fields. The JOIN operator of a symbolic expression will "string" together characters from discontinuous points in memory and allow them to be treated as a unit.

3. GNAMEs are used to reference a continuous section of memory as a unit.

4. DO is the only action operator which can reference an Autocoder pseudo table. This restriction is made because Autocoder pseudo tables are always closed tables, and only the DO operator is used to reference closed tables.

5. Control is resumed by the decision table which referenced the closed decision table. The action immediately following (in the same rule) the DO statement which referenced the closed table is the next action performed.

IBM 7080 **DECISION TABLE LOGIC SHEET**

PROGRAM IDENTIFICATION

PAGE	LINE	CC	TYPE	NO. RULES	ELSE	TABLE NO.	IDENTIFICATION	TABLE NAME	NEXT TABLE NAME	ERROR TABLE NAME
	000	0	N	05	05	0005	TABLE	VALIDATE		

PAGE	LINE	CC		IDENTIFICATION	ORDER OF CONDITION ROWS
	000	1		CONDORDER	

PAGE	LINE	CC		IDENTIFICATION	ORDER OF RULES
	000	2		RULEORDER	0203010405

PROGRAM ______ DATE ______

PREPARED BY ______ PAGE ______

PAGE	LINE	CC		IDENTIFICATION	COMMENT
	000	3		TITLE	CHECK FOR PERMITTED LEDGER NUMBER/DEPARTMENT CHARGES

NEXT PAGE ______

START NEXT CARD: Dup. Col's. 1–5, Punch B in Col. 6, Skip Col's. 7–33, Dup. Col's. 75–80.

PAGE	LINE	CC	ACTION OPERATOR	NAME 1	OP	NAME 2	RULE 1	RULE 2	RULE 3	RULE 4	RULE 5	RULE 6
	01	A		DEPT	GR			5000	6900	7900		
	02	A		DEPT	LE		4990	6890	7890			
	03	A		GENLEDGER	GR		50	60	70	80		
	04	A		GENLEDGER	LE		59	68	79	89		
	05	A										
	06	A	DO	EXPENSERPT			X	X	X	X		
	07	A	MOVE	EXPRECORD	TO	ERRORRCD					X	
	08	A	PUT	ERRORFILE							X	
	09	A	GET	EXPENSE			X	X	X	X	X	
	10	A	GO TO	VALIDATE			X	X	X	X	X	
	11	A										

Situation 7, Decision Table

In Situation 7 it is desired that a summary report be printed listing all acceptable expense transactions. The decision table determines if the departmental charges fall into acceptable general ledger number ranges. Also, certain general ledgers are not to be included on the report. Total lines are to be produced for the previous general ledger number when a new general ledger number is processed. The conditions of the table are familiar. The first action row has as its operand EXPENSERPT. This is the name of a closed pseudo table and the DO operator causes a report writing routine to be entered. In this report pseudo table the descriptive language of the 7080 Report/File Writing system is used. The DTS processor passes these statements along to Autocoder III as they are.

A 1403 Spacing Chart is used to illustrate the format of the desired report. The report has a heading of MONTHLY EXPENSE DISTRIBUTION REPORT, and as part of the heading the letters RPT DATE are stated, followed by "Xs" to indicate the digit positions of the variable date. A line is skipped and the column headings, DEPT NO ACCT NO INVOICE NO INVOICE DATE INVOICE AMOUNT are to be printed. The succeeding lines specify the length, format and special characteristics of each of the fields of the detail line.

In this example of a summary report, it is desired that each transaction be listed until a new general ledger number is encountered, IBM

INTERNATIONAL BUSINESS MACHINES CORPORATION

IBM 1403 SPACING CHART

6 Lines per inch

LINE DESCRIPTION — FIELD HEADINGS WORD MARKS

GLUE

```
Line
 3      X   MONTHLY EXPENSE DISTRIBUTION REPORT   RPT DATE XX XX XX
 5          DEPT NO  ACCT NO  INVOICE NO  INVOICE DATE  INVOICE AMOUNT
 6          XXXX     XXXXX    XXXXXXX     XX XX XX      $XX,XXZ.ZZCR
 7          XXXX     XXXXX    XXXXXXX     XX XX XX      $XX,XXZ.ZZCR
 8          XXXX     XXXXX    XXXXXXX     XX XX XX      $XX,XXZ.ZZCR
24          XXXX     XXXXX    XXXXXXX     XX XX XX      $XX,XXZ.ZZCR
26      XXXGEN LEDGER XX                                $XXX,XXZ.ZZCR
45          XXXX     XXXXX    XXXXXXX     XX XX XX      $XX,XXZ.ZZCR
```

FOLD TO HERE

IBM 7080 — DECISION TABLE NAME DESCRIPTION SHEET

PROGRAM ____ DATE ____ PREPARED BY ____ PAGE ____ NEXT PAGE ____

PG	LINE	CC	TYP	TABLE NO.	IDENT	TABLE NAME
	0 0 0 0		R		TABLE	EXPENSERPT

PG	LINE	NAME	CLASS	NUM.	DESCRIPTION	COMMENTS
D			MODE		REPORT	
		EXPENSERPT	DREPT		EXPENSERPT, EXPAREA1, EXPAREA2, X10, P, 54, 70, EOR,	
			PAHDG			
			CARRC		1	
			CONST	42	MONTHLY EXPENSE DISTRIBUTION REPORT	
			CONST	10	RPT DATE	
			WDATE	08	REPORTDATE	

at which time a total amount field will be printed. The characters ***GEN LEDGER are also to be printed, followed by the two digit general ledger number. The length, format, and special characteristics of this total are specified with the appropriate notations in the print positions desired. The first field in the operand portion names the field where the number will be taken from; this is followed by the printing format using conventional RPT notation.

On the Name Description sheet header, the letter "R" for Report is specified under Type, and the table is named EXPENSERPT. MODE (Page-Line D01) must be specified under class at the beginning of a report, and the entry REPORT made in the description columns. Similarly, MODE must be specified under class as the last line of the description sheet followed by the entry AUTOCODER in the description column.

The entry DREPT (D02) in the class column indicates that a report is to be described, and must be given a name to provide the necessary operand name to be referenced by the programmer when calling for a line to be written. The description entry references other parameters of the report definition; these will not be described in this manual.

The entry PAHDG (D03), for page heading, specifies that a line is required at the top of each page. The entries which follow describe the constant and variable information contained in the heading line. The

IBM 7080 — DECISION TABLE NAME DESCRIPTION SHEET

PROGRAM ______
DATE ______
PREPARED BY ______
PAGE ______
NEXT PAGE ______

PG	LINE	C C	T Y P	TABLE NO.	IDENT	TABLE NAME
	0000	0	R		TABLE	EXPENSERPT

IDENT.

PG	LINE	NAME	CLASS	NUM.	DESCRIPTION	COMMENTS
D	01		MODE		REPORT	
	02	EXPENSERPT	DREPT		EXPENSERPT, EXPAR	EA1, EXPAREA2, X10, P, 54, 70, FOR,
	03		PAHDG			
	04		CARRC		1	
	05		CONST	42	MONTHLY EXPE	NSE DISTRIBUTION REPORT
	06		CONST	10	RPT DATE	
	07		WDATE	08	REPORTDATE	
	08		PAHDG			
	09		CARRC		0	
	10		CONST	52	DEPT NO AC	CT NO INVOICE NO INVOICE DATE INV
	11		CONST	11	OICE AMOUNT	
	12		DTAIL			
	13		CARRC			
	14		CONST	06		
	15		RECRD	04	DEPT,	
	16		CONST	05		
	17		RECRD	05	ACCOUNTNO,	
	18		CONST	04		
	19		RECRD	07	INVOICENO,	
	20		CONST	06		
	21		WDATE	08	INVDATE,	
	22		CONST	06		
	23		RECRD	12	INVAMOUNT, #$XX,XXZ.ZZ ¤CR¤	
	24					
	25					

Situation 7, Report Discription

entry CARRC (D04) is made in the class column to specify carriage control. The first position of the description is a 1 to position the paper to print the first line at the top of the page. The entry CONST (D05) is used to define a line segment which will be the same each time the line is produced. The numeric column contains the number of positions in the segment, and the description specifies the words as numbers to be printed. In this example a 42-position title is specified, followed by a 10-position constant with the letters RPT DATE (D06). The next entry D07), WDATE is used where the data to be moved into a line segment may change between object runs, but will be the same throughout each object run--as in this example the date. The operand in the description area indicates where the date value can be obtained. The report writer will automatically move the date to the proper position.

Following the WDATE entry another page head (D08) is specified followed by a carriage control entry--in this case 0, which is the code for double spacing. The column heads are then specified with two CONST entries (D10, D11).

DTAIL (D12) is used to indicate that the description of a detail line will follow. Succeeding entries describe the fields in the detail line. Each variable field that is to be printed has the field name from where the value is to be taken, followed by the format in which it is to be printed expressed in the operand description area. The fields are placed in the order in which they should appear on the report line. Appropriate spacing

IBM 7080 DECISION TABLE NAME DESCRIPTION SHEET

PG	LINE	C C	T Y P	TABLE NO.	IDENT	TABLE NAME
	0 0 0	0	R		TABLE	EXPENSERPT

PROGRAM ______
DATE ______
PREPARED BY ______
PAGE ______
NEXT PAGE ______

IDENT

PG	LINE	NAME	CLASS	NUM.	DESCRIPTION	COMMENTS
D			MODE		REPORT	
		EXPENSERPT	DREPT		EXPENSERPT, EXPAREA1, EXPAREA2, X10, P, 54, 70, EOR,	
			PAHDG			
			CARRC		1	
			CONST	42	MONTHLY EXPENSE DISTRIBUTION REPORT	
			CONST	10	RPT DATE	
			WDATE	08	REPORTDATE	
			PAHDG			
			CARRC		0	
			CONST	52	DEPT NO ACCT NO INVOICE NO INVOICE DATE INV	
			CONST	11	OICE AMOUNT	
			DTAIL			
			CARRC			
			CONST	06		
			RECRD	04	DEPT,	
			CONST	05		
			RECRD	05	ACCOUNTNO,	
			CONST	04		
			RECRD	07	INVOICENO,	
	20		CONST	06		
			WDATE	08	INVDATE,	
	22		CONST	06		
	23		RECRD	12	INVAMOUNT, #$XX,XXZ.ZZ ¤CR¤	
	24					
	25					

IBM 7080 DECISION TABLE NAME DESCRIPTION SHEET

PG	LINE	C C	T Y P	TABLE NO.	IDENT	TABLE NAME
	0 0 0	0			TABLE	

PROGRAM ______
DATE ______
PREPARED BY ______
PAGE ______
NEXT PAGE ______

IDENT

PG	LINE	NAME	CLASS	NUM.	DESCRIPTION	COMMENTS
E			BREAK	01		
				02	GENLEDGER	
			CARRC		0	
			CONST	16	*** GEN LEDGER	
			RECRD	02	GENLEDGER	
			CONST	33		
			TOTAL	14	INVTOTLEDG, #$XXX,XXZ.ZZ ¤CR¤	
			CLRPT			
			MODE		AUTOCODER	

PG	LINE	NAME	CLASS	NUM.	DESCRIPTION	COMMENTS
		EXPENSE	FILE		TAPE 3003, 2009	FIXED LENTH RECORDS
		EXPRECORD	RNAME			
		ACCOUNTNO	GNAME		SUBLEDGER	
		GENLEDGER		2	N	
		SUBLEDGER		3	N	
		DEPT		4	N	
		INVOICENO		7	A	
		INVDATE		8	A	
		INVAMOUNT		7	#05.02	
		EXPENSERM	PRE	1	#	

and special characters is specified with the CONST entry.

Following the detail line definitions, the entry BREAK (E01) appears in the class column. This, as with the other entries, is normal Report File Writing language. It is a conditional line in the body of the report which permits testing for changes in the contents of specified control fields--in this case a change in the general ledger number. Total lines are automatically written when this control break occurs.

Entries following the BREAK, specify the format of the total line. The entry CLRPT (E08) terminates the report specification. As mentioned previously, the final entry (E09) must be MODE, followed by the word AUTOCODER in the description column.

The report writer pseudo table creates a routine which is essentially a subroutine within the object program. Other programming is required beside the Report statements. In Situation 7 these requirements include:

- o Input/Output procedures -- All data fields processed by the Report/File Writing routine are referenced in the routine as named operands. The standard Decision Table Name Description sheet is used for the description of records contained in the file. The file, itself, is described on a 7080 IOCS File Table Macro-Instruction form

IBM 7080 — DECISION TABLE NAME DESCRIPTION SHEET

PROGRAM ______
DATE ______
PREPARED BY ______
PAGE ______
NEXT PAGE ______

PG	LINE	C C	T Y P		TABLE NO.	IDENT	TABLE NAME	
02	0,0,0	0	D		0000	TABLE	DEFINETO.	

PG	LINE	NAME	CLASS	NUM.	DESCRIPTION	COMMENTS
	01		IOCS			
	02		LASN		@500	
	03		IOTA		EXPENSES¤R006¤200¤	
	04		IOTA		EXPENSERPT¤2008¤2009¤	
	05		IOTA		ERROR¤2005¤	
	06		IOTA		END	
	07		LASN		@25000	
	08					
	09					
	10					
	11					
	12					
	13					
	14					
	15					
	16					
	17					
	18					
	19					
	20					
	21					
	22					
	23					
	24					
	25					

IBM 7080 — DECISION TABLE NAME DESCRIPTION SHEET

PROGRAM ______
DATE ______
PREPARED BY ______
PAGE ______
NEXT PAGE ______

PG	LINE	C C	T Y P		TABLE NO.	IDENT	TABLE NAME	
	0,0,0	0				TABLE		

PG	LINE	NAME	CLASS	NUM.	DESCRIPTION	COMMENTS
	01	EXPENSERPT	FILE			FIXED LENTH 10/ BLOCK
	02	EXPAREA1	RNAME		ØEXPRPTEND	70 CHAR RECORDS
	03		RCD	50		
	04			650		
	05	EXPRPTENDZ	PRE	1¤		GROUP MARK TO STOP WRITING
	06		FILE			SECOND RECORD AREA FOR REPORT WRITER
	07	EXPAREA2	RNAME		ØEXPRPTEND2	
	08		RCD	60		
	09			650		
	10	EXPRPTEND2	PRE	1¤		
	11					
	12					
	13					
	14					
	15					
	16					
	17					
	18					
	19					
	20					
	21					
	22					
	23					
	24					
	25					

FORM X22-6913 PRINTED IN U. S. A.

7080 IOCS FILE TABLE MACRO-INSTRUCTIONS (LONG FORM)

PROGRAM: ______ CODED BY: ______ DATE: ______

EXPENSERPT IOFTA EXPENSERPT □ 000 □ IORETURNTO □ IORETURNTO □ IORETURNTO □

HI / HI / LO / LO □ 0 / 1 / 2 / 3 / 4 □ RWD1 / RWD2 / RUN1 / RUN2 □ EORb / EOFb / BOTH / NONE □ STANDARD / NONSTAND / NOHEADER □ TMbb / NOTM □ STANDARDb / SPECSTANb / NONSTANDb / NOTRAILER □ HSKRWD / NORWDb □

PRIME / NOPRI □ SINGLE / MULTIb □ SEQUEN / NONSEQ □ CKPTRCDSb / NOCKPTRCD □ DUMP / NODP □ DELAYOPEN / NODELAYbb □

IOFTB IP / OP □ DATA / WORK / PROG □ 8 □ INITb / STACK □ CKLNG / NOCHK □ CMPCK / NOCMP □ 2 □ GENAREA / AREAGIV □ F / V □ 0070 □ 0900 □

GETb / PUTb / BOTH □ EXPSERCD / NOWORKAREA □ PAD / NOPA / 00 / NONE □ BOTH / DELET / EOF9S / NONE □ BSDb / WORK / WBSD / NONE □

IOFTC 2 □ CKLNG / NOCHK □ CMPCK / NOCMP □ GP / NO □ F / V □ EXPAREA1 □ EXPAREA2 □

Page	Line	Type	No. Reruns	Else	Table No.	Identification	Table Name	Next Table Name	Error Table Name
0000			01		0004	TABLE	START	TAB0005	

Page	Line	C	Action Operator	Name 1	OP	Name 2	Rule 1	Rule 2	Rule 3	Rule 4	Rule 5	Rule 6
	01	A	OPEN				X					
	02	A	GET	EXPENSE			X					
	03	A										

IBM 7080 DECISION TABLE LOGIC SHEET

PROGRAM IDENTIFICATION

Page	Line	Type	No. Reruns	Else	Table No.	Identification	Table Name	Next Table Name	Error Table Name
0000		M	05	05	0005	TABLE	VALIDATE		

Page	Line	C	Identification	Order of Condition Rows
0001			CONDORDER	

Page	Line	C	Identification	Order of Rules
0002			RULEORDER	0203010405

PROGRAM ______ DATE ______

PREPARED BY ______ PAGE ______

NEXT PAGE ______

Page	Line	C	Identification	Comment
0003			TITLE	CHECK FOR PERMITTED LEDGER NUMBER/DEPARTMENT CHARGES

START NEXT CARD: Dup. Col's. 1-5, Punch 8 in Col. 6, Skip Col's. 7-33, Dup. Col's. 75-80.

Page	Line	C	Action Operator	Name 1	OP	Name 2	Rule 1	Rule 2	Rule 3	Rule 4	Rule 5	Rule 6
	01	A		DEPT	GR			5000	6900	7900		
	02	A		DEPT	LE		4990	6890	7890			
	03	A		GENLEDGER	GR		50	60	70	80		
	04	A		GENLEDGER	LE		59	68	79	89		
	05	A										
	06	A	DO	EXPENSERPT			X	X	X	X		
	07	A	MOVE	EXPRECORD	TO	ERRORRCD					X	
	08	A	PUT	ERRORFILE							X	
	09	A	GET	EXPENSE			X	X	X	X	X	
	10	A	GO TO	VALIDATE			X	X	X	X	X	
	11	A										

Situation 7, Decision Table

Page	Line	Type	No. Reruns	Else	Table No.	Identification	Table Name	Next Table Name	Error Table Name
0000		L	01			TABLE	ENDOFJOB		

Page	Line	C	Action Operator	Name 1	OP	Name 2	Rule 1	Rule 2	Rule 3	Rule 4	Rule 5	Rule 6
	01	A	MOVE	BLANK	TO	GENLEDGER	X					
	02	A	PUT	EXPENSERPT			X					
	03	A	TYPE	'END JOB'			X					
	04	A	CLOSE	EXPENSE			X					
	05	A	CLOSE	EXPENSERPT			X					
	06	A	CLOSE	ERROR			X					
	07	A	GO TO	TABSTOP			X					
	08	A										

(long or short). An IOCS File Table for Situation 7 is shown, this is preceded by Name Description sheets which contain input/output tape assignments (IOTA) and followed by descriptions of the records referenced by the Report/File Writing routine.

- Entrance to the routine -- Decision table 0005 which has been given the name VALIDATE has as its first action a DO command which causes the report pseudo table to be entered. The line will then be printed and control will return to the second action of the rule which had the DO.
- Opening the files -- Table 0004 which has been given the name START causes all files to be opened.
- Closing the files -- The table named ENDOFJOB performs the necessary functions for ending the job. The table name is specified on the IOCS File Table for the EXPENSE input file under END OF FILE TR ADDRESS of IOFTA. After the last record has been read and End of File has been reached, a transfer will go to table ENDOFJOB. As the sample shows, the total is printed for the last general ledger, "END JOB" is typed on the console typewriter, all files are closed and the

PAGE	LINE	CC	TYPE	NO. RULES	ELSE	TABLE NO.	IDENTIFICATION	TABLE NAME	NEXT TABLE NAME	ERROR TABLE NAME
	000	0		01		0004	TABLE	START	TAB0005	

PAGE	LINE	C	ACTION OPERATOR	NAME 1	OP	NAME 2	RULE 1	RULE 2	RULE 3	RULE 4	RULE 5	RULE 6
	01	A	OPEN				X					
	02	A	GET	EXPENSE			X					
	03	A										

IBM 7080

DECISION TABLE LOGIC SHEET

PROGRAM IDENTIFICATION

PAGE	LINE	CC	TYPE	NO. RULES	ELSE	TABLE NO.	IDENTIFICATION	TABLE NAME	NEXT TABLE NAME	ERROR TABLE NAME
	000	0	M	05	05	0005	TABLE	VALIDATE		

PAGE	LINE	CC	IDENTIFICATION	ORDER OF CONDITION ROWS
	000	1	CONDORDER	

PAGE	LINE	CC	IDENTIFICATION	ORDER OF RULES
	000	2	RULEORDER	0203010405

PROGRAM ______ DATE ______

PREPARED BY ______ PAGE ______

NEXT PAGE ______

PAGE	LINE	CC	IDENTIFICATION	COMMENT
	000	3	TITLE	CHECK FOR PERMITTED LEDGER NUMBER/DEPARTMENT CHARGES

START NEXT CARD: Dup. Col's. 1-6, Punch 8 in Col. 6, Skip Col's. 7-23, Dup. Col's. 75-80.

PAGE	LINE	C	ACTION OPERATOR	NAME 1	OP	NAME 2	RULE 1	RULE 2	RULE 3	RULE 4	RULE 5	RULE 6
	01	A		DEPT	GR			5000	6900	7900		
	02	A		DEPT	LE		4990	6890	7890			
	03	A		GENLEDGER	GR		50	60	70	80		
	04	A		GENLEDGER	LE		59	69	79	89		
	05	A										
	06	A	DO	EXPENSERPT			X	X	X	X		
	07	A	MOVE	EXPRECORD		TOERRORRCD					X	
	08	A	PUT	ERRORFILE							X	
	09	A	GET	EXPENSE			X	X	X	X	X	
	10	A	GO TO	VALIDATE			X	X	X	X	X	
	11	A										

Situation 7, Decision Table

PAGE	LINE	CC	TYPE	NO. RULES	ELSE	TABLE NO.	IDENTIFICATION	TABLE NAME	NEXT TABLE NAME	ERROR TABLE NAME
	000	0	N	01			TABLE	ENDOFJOB		

PAGE	LINE	C	ACTION OPERATOR	NAME 1	OP	NAME 2	RULE 1	RULE 2	RULE 3	RULE 4	RULE 5	RULE 6
	01	A	MOVE	BLANK		TOGENLEDGER	X					
	02	A	PUT	EXPENSERPT			X					
	03	A	TYPE	'END JOB'			X					
	04	A	CLOSE	EXPENSE			X					
	05	A	CLOSE	EXPENSERPT			X					
	06	A	CLOSE	ERROR			X					
	07	A	GO TO	TABSTOP			X					
	08	A										

program is halted by the GO TO TABSTOP action.

QUESTIONS

1. In Situation 6, Autocoder was introduced as another language called by the decision tables. Situation 7 calls upon another language, Reports/File Writer. Since the 7080 Decision Table System has its own language why are these languages intermixed?

2. What happens when a Reports/File Writer routine is called?

Answers - Situation 7

1. Decision tables provide a descriptive representation of complex decision procedures in a way that is easy to develop, visualize, and follow through. When a set of procedures does not involve decision logic, the Autocoder language may be a more natural form of expression. The report writer language is a highly developed, but highly specialized language and is used as an auxiliary mode to perform these report writing functions.

2. The report routine is entered and a detail line is written. If a control break occurs, the appropriate total lines will be written automatically. If the last line on the page is written, the report routine will automatically skip to the next page and write the heading information.

IBM 7080 **DECISION TABLE LOGIC SHEET**

PROGRAM IDENTIFICATION: SAMPLE

Page	Line	CC	Type	No. Rules	Else	Table No.	Identification	Table Name	Next Table Name	Error Table Name
BC	000	0	L	07		Ø031	TABLE	SCANLOOP	PRINTIT	TABSTOP

Page	Line	CC	Identification	Order of Condition Rows
	000	1	CONDORDER	

Page	Line	CC	Identification	Order of Rules
BC	000	2	RULEORDER	Ø2Ø3Ø4Ø5Ø6Ø1

Page	Line	CC	Identification	Comment
BC	000	3	TITLE	LOOP TO REPLACE ‡ AND ‡ IN 52 COL. OF INPUT CARD

PROGRAM SAMPLE DATE 5-3-62

PREPARED BY HTH PAGE 10

NEXT PAGE 11

START NEXT CARD: Dup. Col's. 1–5, Punch B in Col. 6, Skip Col's. 7–33, Dup. Col's. 75–80.

Page	Line	C	Action Operator	Name 1	Op	Name 2	01	02	03	04	05	06	07
BC	01	A		FIRSTPASS	IS	ON	Y	N	N	N	N	N	N
BC	02	A		COL23,X2	EQ	'‡'		N	N	Y	Y	N	N
BC	03	A		COL23,X2	EQ	'‡'		N	N			Y	Y
BC	04	A		COUNTER	EQ	+0		N	Y	N	Y	N	Y
	05	A											
BC	06	A	SET	COL23,X2	EQ	'Ø'				X	X		
BC	07	A	SET	COL23,X2	EQ	'0'						X	X
BC	08	A	SET	COUNTER	-	+1		X		X		X	
BC	09	A	SET	COUNTER	EQ	+52	X						
BC	10	A	SET	X2	+	1		X		X		X	
BC	11	A	SET	X2	EQ	COL23	X						
BC	12	A	SET	FIRSTPASS	EQ	ON			X		X		X
BC	13	A	SET	FIRSTPASS	EQ	OFF	X						
BC	14	A	GO TO	SCANLOOP			X	X		X		X	
BC	15	A	GO TO	PRINTIT					X		X		X
	16	A											

Situation 8, Decision Table

Situation 8 illustrates an address modification feature of the 7080 Decision Table System. Address modification utilizes the indirect addressing hardware facility of the 7080. The 7080 DTS supplies the programmer with ten tagged pre-defined address constants (ADCON's) which are used to hold the machine address to be modified. Although these are not described by the user since they are part of the DTS, their specification is shown below:

NAME (6–15)	CLASS (16–20)	NUM. (21–22)	DESCRIPTION (23–38)	CC (39)
XØ	ADCON		@Ø	TEN PRE-TAGGED
X1	ADCON		@Ø	ADCONS PROVIDED
X2	ADCON		@Ø	BY DTS. REFERRED
⋮	⋮		⋮	TO BY X0, X1, . . . X9.
X9	ADCON		@Ø	

The problem which illustrates this address modification feature is also a portion of the DTS processor. The situation involves a loop used in editing input cards to replace non-printing symbols.

In the decision table header, the page is numbered BC. Alphabetics are permissible as page numbers, providing that they are in ascending order for successive tables. Note that in the "Order of Rules" in the decision table header, the rules have been reordered. This has been done to improve the efficiency of the object program. Rules least likely to fail are placed leftmost. A "Comment" is made in this table

IBM 7080

DECISION TABLE LOGIC SHEET

PROGRAM IDENTIFICATION SAMPLE

PAGE	LINE	TABLE NO.	IDENTIFICATION	TABLE NAME	NEXT TABLE NAME	ERROR TABLE NAME
BC	0000	L07	Ø031	TABLE	SCANLOOP PRINTIT	TABSTOP

PAGE	LINE	IDENTIFICATION	ORDER OF CONDITION ROWS
	0001	CONDORDER	

PAGE	LINE	IDENTIFICATION	ORDER OF RULES
BC	0002	RULEORDER	O20304050601

PROGRAM SAMPLE DATE 5-3-62

PREPARED BY HTH PAGE 10

NEXT PAGE 11

PAGE	LINE	IDENTIFICATION	COMMENT
BC	0003	TITLE	LOOP TO REPLACE ‡ AND ǂ IN 52 COL. OF INPUT CARD.

START NEXT CARD: Dup. Col's. 1–5, Punch B in Col. 6, Skip Col's. 7–33, Dup. Col's. 75–80.

PAGE	LINE	C	ACTION OPERATOR	NAME 1	OP	NAME 2	01	02	03	04	05	06	07
BC	01	A		FIRSTPASS	IS	ON	Y	N	N	N	N	N	N
BC	02	A		COL23, X2	EQ	'‡'		N	N	Y	Y	N	N
BC	03	A		COL23, X2	EQ	'ǂ'		N	N			Y	Y
BC	04	A		COUNTER	EQ	+0		N	Y	N	Y	N	Y
	05	A											
BC	06	A	SET	COL23, X2	EQ	'@'				X	X		
BC	07	A	SET	COL23, X2	EQ	'0'						X	X
BC	08	A	SET	COUNTER	-	+1		X		X		X	
BC	09	A	SET	COUNTER	EQ	+52	X						
BC	10	A	SET	X2	+	1		X		X		X	
BC	11	A	SET	X2	EQ	COL23	X						
BC	12	A	SET	FIRSTPASS	EQ	ON			X		X		X
BC	13	A	SET	FIRSTPASS	EQ	OFF	X						
BC	14	A	GO TO	SCANLOOP			X	X		X		X	
BC	15	A	GO TO	PRINTIT					X		X		X
	16	A											

Situation 8, Decision Table

header. It is permissible to make any remarks the programmer cares to make here, and in this example an explanation of the decision table's function is given: "LOOP TO REPLACE ‡ AND ǂ IN 52 COL. OF INPUT CARD."

The object of the program is to replace the non-printing symbols which might appear in columns 23 to 74 of the card. Hence each column must be examined for the presence or absence of either of the two symbols.

The first condition row checks to see if a Bit switch (a one character constant set up in the program as a switch) is ON or OFF. If rule 1 is satisfied, i.e., the Bit switch is found to be ON, the scanning process is to begin. The first action specified for rule 1 sets a counter to 52. The second action sets the Address Register X2 to COL23, i.e. the machine address of column 23 of the card. The third action sets the Bit switch to OFF and the final action calls for a reiteration of the loop.

In rule 2, the bit switch is found to be OFF meaning that this is not the starting point. The second and third conditions determine that the non-printing characters do not occur in the column currently being examined.

Consider the first operand of these condition statements-- COL23,X2. The Address Register X2 contains the actual address of the character in question (set in Rule 1), and COL23 provides the format of the field (1 position alphameric). This is the construction of operands

IBM 7080

DECISION TABLE LOGIC SHEET

PROGRAM IDENTIFICATION SAMPLE

PAGE	LINE	CC	TYPE	NO. RULES	FILE	TABLE NO.	IDENTIFICATION	TABLE NAME	NEXT TABLE NAME	ERROR TABLE NAME
BC	000	0	L	07		ØC31	TABLE	SCANLOOP	PRINTIT	TABSTOP

PAGE	LINE	CC		IDENTIFICATION	ORDER OF CONDITION ROWS
	000	1		CONDORDER	

PAGE	LINE	CC		IDENTIFICATION	ORDER OF RULES
BC	000	2		RULEORDER	020304050601

PROGRAM SAMPLE DATE 5-3-62

PREPARED BY HTH PAGE 10

NEXT PAGE 11

PAGE	LINE	CC		IDENTIFICATION	COMMENT
BC	000	3		TITLE	LOOP TO REPLACE ‡ AND ≢ IN 52 COL. OF INPUT CARD.

START NEXT CARD: Dup. Col's. 1–5, Punch B in Col. 6, Skip Col's. 7–33, Dup. Col's. 75–80.

PAGE	LINE	C	ACTION OPERATOR	NAME 1	OP	NAME 2	01	02	03	04	05	06	07
BC	01	A		FIRSTPASS	IS	ON	Y	N	N	N	N	N	N
BC	02	A		COL23, X2	EQ	'‡'		N	N	Y	Y	N	N
BC	03	A		COL23, X2	EQ	'≢'		N	N			Y	Y
BC	04	A		COUNTER	EQ	+0		N	Y	N	Y	N	Y
	05	A											
BC	06	A	SET	COL23, X2	EQ	'@'				X	X		
BC	07	A	SET	COL23, X2	EQ	'◊'						X	X
BC	08	A	SET	COUNTER	–	+1		X		X		X	
BC	09	A	SET	COUNTER	EQ	+52	X						
BC	10	A	SET	X2	+	1		X		X		X	
BC	11	A	SET	X2	EQ	COL23	X						
BC	12	A	SET	FIRSTPASS	EQ	ON			X		X		X
BC	13	A	SET	FIRSTPASS	EQ	OFF	X						
BC	14	A	GO TO	SCANLOOP			X	X		X		X	
BC	15	A	GO TO	PRINTIT					X		X		X
	16	A											

Situation 8, Decision Table

used for address modification. To get the next character of the card X2 will be increased by 1. The fourth condition determines that the end has not been reached.

The actions then call for (1) decrementing the counter by 1, (2) incrementing the column address in the Address Register X2 by 1, and (3) reiterating the loop, so that the next column will be examined.

The third rule determines that the Bit switch is not ON and the non-printing characters do not occur in the column being examined, but in the fourth condition row it is discovered that the last column has been examined. The actions therefore call for a resetting of the Bit switch so that on the next iteration of the loop rule 1 (the starting condition) will be executed. Since the last column has been examined, the only other action called for is the execution of another table: PRINTIT, which will print the line.

Rules 4, 5, 6, and 7 each encounter non-printing characters and make the appropriate substitutions.

In this situation by merely incrementing X2, each of the 52 columns are examined, and the character in the storage position replaced if either the character ǂ or ⧻ occurs. Termination and printing occurs when the counter reaches zero after being decremented by one for each of the 52 columns on the card.

QUESTIONS

1. How much programming effort is involved in address modification utilizing the 7080 Decision Table System?
2. What is a bit switch (BITSW) and what is its function?

Answers - Situation 8

1. Very little. The 7080 DTS supplies the programmer with ten named and pre-defined address constants which are used to hold an address to be modified. The only programming required is initializing, incrementing or decrementing these. Address Registers.

2. One character variables are set up and their 1, 2, 4 and/or A bits named. Each named bit can then be used by the program as an ON/OFF switch; i. e., the program may set it to the ON or OFF condition and at other points in the program its state can be tested.

IBM 7080 — DECISION TABLE NAME DESCRIPTION SHEET

PROGRAM ______ DATE ______ PREPARED BY ______ PAGE ______ NEXT PAGE ______

PG	LINE	CC	TYP	TABLE NO.	IDENT	TABLE NAME
	0.0.0	0			TABLE	

PG	LINE	NAME	CLASS	NUM.	DESCRIPTION	COMMENTS
	01					
	02					
	03	ASSEMBLY	FILE		TAPE 2001, 2002	VARIABLE LENTH RECORDS UNBLOCKED
	04		RNAME		ASMBLYEND	1000 CHAR MAX
	05	ASMBLYID	FLD	12A		
	06	ASMBLYTIME		2N		
	07					
	08					
	09	TOTALPARTS		2N		
	10	STARTVAR		1A		
	11			963A		
	12	ASMBLYEND		1A		
	13					

Situation 9, Figure 9A

IBM 7080 — DECISION TABLE NAME DESCRIPTION SHEET

PROGRAM ______ DATE ______ PREPARED BY ______ PAGE ______ NEXT PAGE ______

PG	LINE	CC	TYP	TABLE NO.	IDENT	TABLE NAME
03	0.0.0	0	D	0002	TABLE	MISCVARBLE

PG	LINE	NAME	CLASS	NUM.	DESCRIPTION	COMMENTS
	01	SIZE	FLD	2+		USED TO HOLD ITEM LENGTH
	02					
	03	FIRSTITEM	ADCON		STARTVAL	LOC OF 1ST CHAR OF VARIABLE PORTION
	04					
	05	INDXA	POINT		HI	HIGH SPEED INDEX POINTER
	06					

Situation 9, Figure 9B

PAGE	LINE	CC	TYPE	NO. RULES	ELSE	TABLE NO.	IDENTIFICATION	TABLE NAME	NEXT TABLE NAME	ERROR TABLE NAME
	0000	0	1	01			TABLE	NEWASMBLY		

PAGE	LINE	CC	ACTION OPERATOR	NAME 1	OP	NAME 2	RULE 1	RULE 2	RULE 3	RULE 4	RULE 5	RULE 6
	01	A	GET	ASSEMBLY			X					
	02	A	SET	INDXA	EQ	FIRSTITEM	X					
	03	A	MOVEV	INDXA, 5	TO	NEXTASMBLY	X					
	04	A	GO TO	PREPORDER			X					
	05	A										

Situation 9, Figure 9C

Situation 9 is concerned with producing purchase orders, shop orders, and stock orders for the various parts which make up a product assembly. The file which contains the manufacturing information about a particular assembly is composed of variable length records -- a fixed portion of general information followed by a variable number of variable size items pertaining to each part which makes up the assembly. These variable items are referred to as trailer items.

The problem now arises of calling these items from the file since the length of the next item is not known.

The assembly file shown in Figure 9 A is shown with certain fixed header information: ASMBLYID, ASMBLYTIME ... TOTALPARTS. The variable portion starts with STARTVAR, a one position field. This name of the first character of the variable portion is used as the operand of the ADCON in Figure 9B.

The ADCON named FIRSTITEM will contain the 7080 address of the first character position of the variable portion of the record. The ADCON named FIRST ITEM is used to initialize the index pointer INDXA, such that information can later be moved out of the variable record into the NEXT-ASMBLY working storage area.

In the first decision table, Figure 9C, GET NEWASMBLY brings in the first record from the file named ASSEMBLY. The second action sets the index pointer, INDXA, equal to the address of the first character in the variable length position of the input record. The third action

IBM 7080

DECISION TABLE NAME DESCRIPTION SHEET

PROGRAM ____
DATE ____
PREPARED BY ____
PAGE ____
NEXT PAGE ____

PG	LINE	CC	TYP	TABLE NO.	IDENT	TABLE NAME
02	0000	0	D	0001	TABLE	WORKAREA

IDENT.

PG	LINE	NAME	CLASS	NUM.	DESCRIPTION	COMMENTS
02			FILE	03	THREE TYPES OF	ITEMS WILL BE OVERLAPPED
		NEXTASMBLY	RNAME	O	MFGDEPT	
		MANUFACTUR	GNAME		MFGDEPT	DEFINE MAKE ITEM
		PARTTYPE	BITCD			ONLY ONE BIT ON AT A TIME
		BUY		1		
		MAKE		2		
		SUBASMBLY		4		
		LASTITEM		A		END OF VARIABLE LENGTH RECORD
		PARTNUMBER	FLD	9N		
		QUANTITY		2N		
		MFGDEPT		8A		
	12	PURCHASE	RNAME		GROSSLIMIT	DEFINE BUY ITEM
	13		FLD	1A		
	14	PURCHASENO		13N		
	15	PARTCLASS		2N		
	16	DESCRIPTION		30A+		
	17	DELIVERTIM		2N		
	18	ORDERAMT		4N		
	19	UNITPRICE		7#+05.02		
	20	GROSSLIMIT		6#+05.01		
	21	SUBASBLYRQ	RNAME		STOCKLOC	DEFINE SUBASSEMBLY ITEM
	22		FLD	1A		
	23	ASMBLYNAME		12A		
	24	QTYREQRD		2N		
	25	STOCKLOC		10A		

Situation 9, Figure 9D

MOVEV INDXA, 5 TO NEXTASMBLY causes 5 characters to be transmitted to the working storage area. The transmission begins at the location contained in INDXA. The final action transfers control to a table named PREPORDER.

Note that there is only one rule to this table. Such tables are called "unconditional" tables.

The first five characters (a minimum with the high-speed transmit called for in the POINT entry) of the variable portion of the record now reside in the working storage area named NEXTASMBLY. This area is described in Figure 9D. Notice that the FILE class entry has a 3 in the numeric portion. This indicates that the area is to be described or redefined in three different ways: one for each possible format. After the first, subsequent RNAMEs have the effect of a LASN back to the original RNAME; i.e., the same area may now be redescribed. The first RNAME in this example is NEXTASMBLY. The first character is defined as a Bit code (BITCD). The 1-Bit means BUY the part, the 2-Bit means MAKE the part, the 4-Bit means order a SUBASMBLY (sub-assembly) from the stock room; the A-Bit only goes ON for the last character in the variable length record to indicate that there are no more items. The various other fields are then described for this item. The second RNAME called PURCHASE is then overlapped on the first area. The format of this item is defined and names are associated with

IBM 7080

DECISION TABLE LOGIC SHEET

PROGRAM IDENTIFICATION

PAGE	LINE	CC	TYPE	NO. RULES	ELSE	TABLE NO.	IDENTIFICATION	TABLE NAME	NEXT TABLE NAME	ERROR TABLE NAME
00	00					M0505	TABLE	PREPORDER		

PAGE	LINE	CC	IDENTIFICATION	ORDER OF CONDITION ROWS
00	01		CONDORDER	

PAGE	LINE	CC	IDENTIFICATION	ORDER OF RULES
00	02		RULEORDER	

PAGE	LINE	CC	IDENTIFICATION	COMMENT
00	03		TITLE	

PROGRAM ______ DATE ______

PREPARED BY ______ PAGE ______

NEXT PAGE ______

START NEXT CARD: Dup. Col's. 1–5, Punch B in Col. 6, Skip Col's. 7–33, Dup. Col's. 75–80.

PAGE	LINE	C	ACTION OPERATOR	NAME 1	OP	NAME 2	RULE 1	RULE 2	RULE 3	RULE 4	RULE 5	RULE 6
	01	A										
	02	A										
	03	A		ON	IS		BUY	MAKE	SUBASMBLY	LASTITEM		
	04	A										
	05	A	SET	SIZE	EQ		BUYSIZE	MAKESIZE	SUBSIZE			
	06	A	MOVEV	INDXA,SIZE	TO	NEXTASMBLY	X	X	X			
	07	A	DO				PURCHORDER	SHOPORDER	STOCKORDER			
	08	A	SET	TOTALPRICE	EQ	FORMULA1	X					
	09	A	SET	INDXA	+	SIZE	X	X	X			
	10	A	MOVEV	INDXA,5	TO	NEXTASMBLY	X	X	X			
	11	A	GO TO				PREPORDER	PREPORDER	PREPORDER	NEWASMBLY	ERRORANALY	
	12	A										
	13	A										
	14	A										
	15	A										
	16	A										
	17	A										
	18	A										
	19	A										
	20	A										
	21	A										
	22	A										
	23	A										
	24	A										
	25	A										

Situation 9, Figure 9E

the various fields such that they can be addressed directly in the working storage area. Finally, the third type of item, SUBASBLYRQ, is overlapped at the start of the work area.

Since the variable portion of the input record will contain a variable number of these items in a random order, they cannot be conveniently defined and addressed in the input area. Recall now that the first five characters have been sent into the working area, and the Bit switch can be tested to determine which type of item has been brought in: PURCHASE, MANUFACTUR, or SUBASBLYRQ. This interrogation is made in the condition area in Figure 9E. Bit switches are tested by the action:

Bit switch name	IS	ON(OFF)
ON(OFF)	IS	Bit switch name

In this application only one Bit may be on at a time. In the decision table 9E, Rule 1 determines if the part should be bought; Rule 2, if the part should be made; and Rule 3, whether a sub-assembly should be ordered. In Rule 4, the A Bit (LAST ITEM) is tested to determine if this is character of the variable portion.

Assume that the BUY Bit of character 1 in the NEXTASMBLY area is ON, and consider the actions of Rule 1. The first two actions move up the rest of the item still in the variable portion of the input area. SET SIZE EQ BUYSIZE moves the value 20 (the length of the purchase item) to SIZE. BUYSIZE is defined in Table 0003 (Figure 9F)

IBM 7080 **DECISION TABLE LOGIC SHEET**

PROGRAM IDENTIFICATION

PAGE	LINE	CC	TYPE	NO. RULES	ELSE	TABLE NO.	IDENTIFICATION	TABLE NAME	NEXT TABLE NAME	ERROR TABLE NAME
	0000			M0505			TABLE	PREPORDER		

PAGE	LINE	CC			IDENTIFICATION	ORDER OF CONDITION ROWS
	0001				CONDORDER	

PAGE	LINE	CC		IDENTIFICATION	ORDER OF RULES
	0002			RULEORDER	

PAGE	LINE	CC		IDENTIFICATION	COMMENT
	0003			TITLE	

PROGRAM ______ DATE ______

PREPARED BY ______ PAGE ______

NEXT PAGE ______

START NEXT CARD: Dup. Col's. 1–5, Punch B in Col. 6, Skip Col's. 7–33, Dup. Col's. 75–80.

PAGE	LINE	CC	ACTION OPERATOR	NAME 1	OP	NAME 2	RULE 1 OP	RULE 1 NAME 2	RULE 2 OP	RULE 2 NAME 2	RULE 3 OP	RULE 3 NAME 2	RULE 4 OP	RULE 4 NAME 2	RULE 5 OP	RULE 5 NAME 2	RULE 6 OP	RULE 6 NAME 2
	01	A																
	02	A																
	03	A		ON	IS			BUY		MAKE		SUBASMBLY		LASTITEM				
	04	A																
	05	A	SET	SIZE	EQ			BUYSIZE		MAKESIZE		SUBSIZE						
	06	A	MOVEV	INDXA, SIZE	TO	NEXTASMBLY	X		X		X							
	07	A	DO					PURCHORDER		SHOPORDER		STOCKORDER						
	08	A	SET	TOTALPRICE	EQ	FORMULA1	X											
	09	A	SET	INDXA	+	SIZE	X		X		X							
	10	A	MOVEV	INDXA, 5	TO	NEXTASMBLY	X		X		X							
	11	A	GO TO					PREPORDER		PREPORDER		PREPORDER		NEWASMBLY		ERRORANALY		
	12	A																
	13	A																
	14	A																
	15	A																
	16	A																
	17	A																
	18	A																
	19	A																
	20	A																
	21	A																
	22	A																
	23	A																
	24	A																
	25	A																

Situation 9, Figure 9E

IBM 7080 **DECISION TABLE NAME DESCRIPTION SHEET**

PG	LINE	CC	TYP	TABLE NO.	IDENT	TABLE NAME
04	0.0.0.0	0	D	0003	TABLE	MISCFIXED

PROGRAM ______

DATE ______

PREPARED BY ______

PAGE ______

NEXT PAGE ______

PG	LINE	NAME	CLASS	NUM.	DESCRIPTION	COMMENTS
	01	FORMULA1	MEXP		#+05.02,ORDERAMT * UNITPRICE	
	02					
	03	BUYSIZE	PCON	2	+20	
	04	MAKESIZE	PCON	2	+65	
	05	SUBSIZE	PCON	2	+25	
	06					
	07	VENDORLIST	RLIST		02,20,VENDORINFO	
	08				01,ABC CORP. 206E 57, NYC	
	09				02-29,,XYZ CO. 1,174 CONN. DC	
	10				50-63,IJC, INC.12 MAIN	
	11				75, ZELCO POST RD LOCAL	
	12					

Situation 9, Figure 9F

as class PCON, meaning permanent constant; i. e., a constant whose value does not change during the running of the program, as opposed to a variable whose value will change. The field names SIZE now contains the value +20. The second action MOVEV INDXA, SIZE TO NEXTASMBLY will cause the transmission of the first 20 characters from the variable area to working storage. The command MOVEV means move variable length.

Action 3 calls for the execution of PURCHORDER, a closed table. Action 4 calls for the evaluation of FORMULA1 (the expression is defined in Table 0003, Figure 9F) and the placing of the resulting value in TOTALPRICE. The next action calls for the incrementing of INDXA by the size of the item that was last moved up. In this case, INDXA would be incremented by 20 in preparation for moving the next five characters to the working storage area. Then it can be determined what the next item might be. Action 6 does just this, it moves five characters from the location specified by INDXA to NEXTASMBLY. The last command of Rule 1 calls for a return to the beginning of the table.

A new Bit switch character is now residing in the first position of the working area and an interrogation can be made to determine what the next item is; i. e., PURCHASE, MANUFACTUR, SUBASBLYRQ, or the last character in the record.

The table PREPORDER is the main processing table of the program.

IBM 7080 — DECISION TABLE LOGIC SHEET

PROGRAM IDENTIFICATION

Page	Line	CC	Type	No. Rules	Else	Table No.	Identification	Table Name	Next Table Name	Error Table Name
	0000	L	64	64			TABLE	PURCHORDERDO		

Page	Line	CC	Identification	Order of Condition Rows
	0001		CONDORDER	

Page	Line	CC	Identification	Order of Rules
	0002		RULEORDER	

Page	Line	CC	Identification	Comment
	0003		TITLE	

PROGRAM ______ DATE ______

PREPARED BY ______ PAGE ______

NEXT PAGE ______

START NEXT CARD: Dup. Col's. 1–5, Punch B in Col. 6, Skip Col's. 7–33, Dup. Col's. 75–80.

Page	Line	CC	Action Operator	Name 1	OP	Name 2	Rule 1: 01	02	03	04	05	06	Rule 2	Rule 3	Rule 4	Rule 5	Rule 6
	01	A		PARTCLASS	IS	NUMERIC	Y	Y	Y								
	02	A		PARTCLASS	EQ	VENDORLIST	Y	Y	N								
	03	A		GROSSPRICE	GE	TOTALPRICE	Y	N									
	04	A															
	05	A	MOVE	VENDORINFO	TO	ADDRESS	X	X									
	06	A	MOVE	DELIVERTIM	TO	WHENNEEDED	X	X									
	07	A	SET	PURCHASEID	EQ	PURCHASEND	X	X		X							
	08	A	MOVE	ORDERAMT	TO	QTY	X	X		X							
	09	A	MOVE	DESCRIPTON	TO	PURCHDECRP	X	X		X							
	10	A	PUT	PURCHCARD			X	X									
	11	A	DO	SPECVENDOR					X								
	12	A	DO	SPECNOTICE				X									
	13	A	DO	SPECCASES						X							
	14	A															

Situation 9, Figure 9G

IBM 7080 — DECISION TABLE NAME DESCRIPTION SHEET

PROGRAM ______

DATE ______

PREPARED BY ______

PAGE ______

NEXT PAGE ______

PG	Line	CC	Typ	Table No.	Ident	Table Name
04	0000	0	D	0003	TABLE	MISCFIXED

PG	Line	Name	Class	Num.	Description	Comments
	01	FORMULA1	MEXP		#405.02,ORDERAMT * UNITPRICE	
		BUYSIZE	PCON		2+20	
		MAKESIZE	PCON		2+65	
		SUBSIZE	PCON		2+25	
		VENDORLIST	RLIST		02,20,VENDORINFO	
					01,ABC CORP. 206E57, NYC	
					02-29,,XYZ CO. 1,174 CONN. DC	
					50-63,JSC INC.12 MAIN	
					75,ZELCO POST RD LOCAL	

Situation 9, Figure 9F

As each item identification is brought up from the variable area, its type is determined and the rest of the item, now that the size is known, is moved up. Aside from certain housekeeping functions, a closed table is executed which will prepare a purchase order for a purchase item, or a shop order for an item which is to be made in the shop, or a stock order for sub-assemblies which are to be delivered from the stock room.

The PURCHORDER table in 9G illustrates what one of these closed tables might look like. Notice that the table is entirely in limited entry form, thus, the narrow two-position entry columns are used.

The second condition demonstrates the use of a new operand-- a reference list. The reference list is composed of arguments and functions (values). (In contrast a VLIST contains only arguments.) A reference list for Situation 9 is shown in Figure 9F; it is called VENDORLIST. A reference list has almost the same properties as a TABLE macro in Autocoder. In the description portion of the first line, the first value indicates the size of the argument, the second value indicates the size of the function and the third entry, a name, specifies where the function is to be placed when a successful search of the table is made. Subsequent lines are used to indicate the argument and function values; these <u>must</u> be the same length as shown in the list

IBM 7080 **DECISION TABLE LOGIC SHEET**

PROGRAM IDENTIFICATION

PAGE	LINE	CC	TYPE	NO. RULES	ELSE	TABLE NO.	IDENTIFICATION	TABLE NAME	NEXT TABLE NAME	ERROR TABLE NAME
	000	0	L	04	64		TABLE	PURCHORDERDO		

PAGE	LINE	CC	IDENTIFICATION	ORDER OF CONDITION ROWS
	000	1	CONDORDER	

PAGE	LINE	CC	IDENTIFICATION	ORDER OF RULES
	000	2	RULEORDER	

PAGE	LINE	CC	IDENTIFICATION	COMMENT
	000	3	TITLE	

PROGRAM ______ DATE ______

PREPARED BY ______ PAGE ______

NEXT PAGE ______

START NEXT CARD: Dup. Col's. 1–5, Punch B in Col. 6, Skip Col's. 7–33, Dup. Col's. 75–80.

PAGE	LINE	CC	ACTION OPERATOR	NAME 1	OP	NAME 2	01	02	03	04	05	06
	01	A		PARTCLASS	IS	NUMERIC	Y	Y	Y			
	02	A		PARTCLASS	EQ	VENDORLIST	Y	Y	N			
	03	A		GROSSPRICE	GE	TOTALPRICE	Y	N				
	04	A										
	05	A	MOVE	VENDORINFO	TO	ADDRESS	X	X				
	06	A	MOVE	DELIVERTIM	TO	WHENNEEDED	X	X				
	07	A	SET	PURCHASEID	EQ	PURCHASIND	X	X		X		
	08	A	MOVE	ORDERAMT	TO	QTY	X	X		X		
	09	A	MOVE	DESCRIPTON	TO	PURCHDECRP	X	X		X		
	10	A	PUT	PURCHCARD			X	X				
	11	A	DO	SPECVENDOR					X			
	12	A	DO	SPECNOTICE				X				
	13	A	DO	SPECCASES						X		
	14	A										

Situation 9, Figure 9G

IBM 7080 **DECISION TABLE NAME DESCRIPTION SHEET**

PROGRAM ______

DATE ______

PREPARED BY ______

PAGE ______

NEXT PAGE ______

PG	LINE	CC	TYP	TABLE NO.	IDENT	TABLE NAME
04	0000	0	D	0003	TABLE	MISCFIXED

PG	LINE	NAME	CLASS	NUM.	DESCRIPTION	COMMENTS
		FORMULA1	MEXP		#405.02,ORDERAMT	* UNITPRICE
		BUYSIZE	PCON	2	+20	
		MAKESIZE	PCON	2	+65	
		SUBSIZE	PCON	2	+25	
		VENDORLIST	RLIST		02,20,VENDORINFO	
					01,ABC CORP. 206	E 17. NYC
					02 29, XYZ CO. 1,1	74 CONN. DC
					50-63, JSC INC. 12	MAIN
					75, ZELCO POST RD	LOCAL

Situation 9, Figure 9F

header line. Note that a range of values; e.g., 02-29 can be used when the values are numeric and in ascending sequence. The second condition of Rule 1, then, checks to see if the value of PARTCLASS, a field in the PURCHASE item, is contained on the reference list. For example, if the value of PARTCLASS is 16, the test will be passed since it is in the range 02-29 and the function XYC CO. 1174 CONN DC would be moved to the field called VENDORINFO.

If the other conditions in Rule 1, Figure 9G, are satisfied, the first action calls for moving this vendor information (the function found in the reference list) to the field called ADDRESS on the purchase order. Subsequent actions call for moving other data to the purchase card and finally PUT PURCHCARD will make this information available to the output unit. Since there are no more actions following this command, control will be returned to the table PREPORDER at action 4 in Rule 1.

Note that while executing this table the program recognizes that the current item in working storage is a purchase item whose fomat has been defined. Consequently, the names and format which apply can be used directly.

The closed table SHOPORDER and STOCKORDER (not shown) will be executed in a like fashion if Rule 2 or 3 is satisfied. The first condition of the PURCHORDER Table is an illustration of a <u>state</u> condition

IBM 7080

DECISION TABLE LOGIC SHEET

PROGRAM IDENTIFICATION

PAGE	LINE	CC	TYPE	NO. RULES	ELSE	TABLE NO.	IDENTIFICATION	TABLE NAME	NEXT TABLE NAME	ERROR TABLE NAME
	0000					M0505	TABLE	PREPORDER		

PAGE	LINE	CC	IDENTIFICATION	ORDER OF CONDITION ROWS
	0001		CONDORDER	

PAGE	LINE	CC	IDENTIFICATION	ORDER OF RULES
	0002		RULEORDER	

PROGRAM ________ DATE ________

PREPARED BY ________ PAGE ________

NEXT PAGE ________

PAGE	LINE	CC	IDENTIFICATION	COMMENT
	0003		TITLE	

START NEXT CARD: Dup. Col's. 1–5, Punch B in Col. 6, Skip Col's. 7–33, Dup. Col's. 75–80.

PAGE	LINE	CC	ACTION OPERATOR	NAME 1	OP	NAME 2	RULE 1	RULE 2	RULE 3	RULE 4	RULE 5	RULE 6
	01	A										
	02	A										
	03	A		ON	IS		BUY	MAKE	SUBASMBLY	LASTITEM		
	04	A										
	05	A	SET	SIZE	EQ		BUYSIZE	MAKESIZE	SUBSIZE			
	06	A	MOVEV	INDXA, SIZE	TO	NEXTASMBLY	X	X	X			
	07	A	DO				PURCHORDER	SHOPORDER	STOCKORDER			
	08	A	SET	TOTALPRICE	EQ	FORMULA1	X					
	09	A	SET	INDXA	+	SIZE	X	X	X			
	10	A	MOVEV	INDXA, 5	TO	NEXTASMBLY	X	X	X			
	11	A	GO TO				PREPORDER	PREPORDER	PREPORDER	NEWASMBLY	ERRORANALY	
	12	A										
	13	A										
	14	A										
	15	A										
	16	A										
	17	A										
	18	A										
	19	A										
	20	A										
	21	A										
	22	A										
	23	A										
	24	A										
	25	A										

Situation 9, Figure 9E

and asks the question--is the value of PARTCLASS all numeric? If it is entirely numeric, the test is passed. If there is a symbol other than the digits 0-9 in this field, the test will fail. Other state conditions are POSITIVE, NEGATIVE, ZERO, BLANK, ON, and OFF. The state term may be either the first or the second operand in a condition statement. The condition in PREPORDER shows the state term ON in the operand one position and the Bit code name in operand 2.

This situation has illustrated a means of using the POINT (in reality a six-position field containing a 7080 address), to work through the variable portion of a record without doing direct address modification. The formats of the various items which may appear in the variable section are defined in a working storage section in an overlapped manner; then, as each new item is moved in, an interrogation is made as to the type of item; and then the names of the fields of the item can be referenced directly. At this point the size of the item is also known and it is possible to increment the index pointer to the beginning of the next item. If the size of the item appears directly in the data itself, this information could also be used to increment the index pointer in working through the data.

The index pointer concept can be used in a like manner on the output side to fabricate variable length output records. In this case, the pointer is used to indicate where the next item is to appear. Since the

IBM 7080 — DECISION TABLE LOGIC SHEET

PROGRAM IDENTIFICATION

PAGE	LINE	CC	TYPE	NO. RULES	ELSE	TABLE NO.	IDENTIFICATION	TABLE NAME	NEXT TABLE NAME	ERROR TABLE NAME
	000	0	M	05	05		TABLE	PREPORDER		

PAGE	LINE	CC	IDENTIFICATION	ORDER OF CONDITION ROWS
	000	1	CONDORDER	

PAGE	LINE	CC	IDENTIFICATION	ORDER OF RULES
	000	2	RULEORDER	

PAGE	LINE	CC	IDENTIFICATION	COMMENT
	000	3	TITLE	

PROGRAM ______ DATE ______

PREPARED BY ______ PAGE ______

NEXT PAGE ______

START NEXT CARD: Dup. Col's. 1–5, Punch B in Col. 6, Skip Col's. 7–33, Dup. Col's. 75–80.

LINE	C	ACTION OPERATOR	NAME 1	OP	NAME 2	RULE 1	RULE 2	RULE 3	RULE 4	RULE 5	RULE 6
01	A										
02	A										
03	A		ON	IS		BUY	MAKE	SUBASMBLY	LASTITEM		
04	A										
05	A	SET	SIZE	EQ		BUYSIZE	MAKESIZE	SUBSIZE			
06	A	MOVEV	INDXA,SIZE	TO	NEXTASMBLY	X	X	X			
07	A	DO				PURCHORDER	SHOPORDER	STOCKORDER			
08	A	SET	TOTALPRICE	EQ	FORMULA1	X					
09	A	SET	INDXA	+	SIZE	X	X	X			
10	A	MOVEV	INDXA,5	TO	NEXTASMBLY	X	X	X			
11	A	GO TO				PREPORDER	PREPORDER	PREPORDER	NEWASMBLY	ERRORANALY	
12	A										
13	A										
14	A										
15	A										
16	A										
17	A										
18	A										
19	A										
20	A										
21	A										
22	A										
23	A										
24	A										
25	A										

Situation 9, Figure 9E

IBM 7080 — DECISION TABLE NAME DESCRIPTION SHEET

PROGRAM ______

DATE ______

PREPARED BY ______

PAGE ______

NEXT PAGE ______

PG	LINE	CC	TYP	TABLE NO.	IDENT	TABLE NAME
03	000	0	D	0002	TABLE	MISCVARBLE

PG	LINE	NAME	CLASS	NUM.	DESCRIPTION	COMMENTS
	01	SIZE	FLD	24		USED TO HOLD ITEM LENGTH
	02					
	03	FIRSTITEM	ADCON		STARTVAL	LOC OF 1ST CHAR OF VARIABLE PORTION
	04					
	05	INDXA	POINT		HI	HIGH SPEED INDEX POINTER
	06					

Situation 9, Figure 9B

size of the current item is often known, this number can be used in conjunction with the index pointer to move the proper amount of information to the output area. The pointer is then incremented by that value to be ready for the next item to be placed in the output area.

ADCONs are generally used to initialize the index pointers to the first position of the variable portion of an input or output record.

It is also possible to use an index pointer to move all the data up to the record mark terminating the record. The format for this variation is:

MOVE index pointer name, RM TO work area name

A high-speed transmit will be assumed in this case. Recall that in the definition of POINT the operand field contained HI, this indicates that high-speed transmit will be used when this index pointer is used. If increments of other than five characters are to be moved, SER will call for a serial transmit moving data into or out of a variable length record.

QUESTIONS

1. How is an index pointer initialized to the address of the first character of a variable length record?
2. How can variable length fields in random order be addressed by name?
3. What is an RLIST?
4. How is an index pointer used in connection with output?

Answers - Situation 9

1. The first position of the input is assigned a name on the Name Description sheet. This name is then used as the operand for an ADCON. The name of the ADCON is then used to initialize the index pointer with a SET... EQ action operator.

2. By describing a working storage area in an overlapped manner for each of the possible data formats and then testing a field of the trailer item to determine which type of item is present.

3. A reference list is a series of arguments and functions (values), used to provide function values based on a certain input argument. In the example, if PARTCLASS is 01, the function is ABC CORP. 206E57 NYC.

4. The index pointer concept is used to fabricate variable length output records. The programmer is responsible for maintaining the proper value of the pointer.

CHAPTER 3

Sample Program Using 7080 DTS

The various capabilities of the DTS are demonstrated in the sample problem below. The program is not a complete job: the data definition is thorough but all the tables to handle the entire payroll application are not shown. For example, many of the fields defined are not referenced in the procedure portion of the program. Nevertheless, the sample program will illustrate most of the features of the language for both defining the data and the processing.

The sample program was written to serve three purposes, (1) test the language to provide feedback on the adequacy of the language, (2) provide a test problem to check out the processor and (3) as an illustration to be used in this manual.

To illustrate how a 7080 DTS program looks at various points in time, different formats are used to show the program. The first few pages show the sheets used to initially prepare the data definition, including the Decision Table Name Description sheet and the 7080 IOCS File Macro-Instruction sheet. The remainder of the data definition is shown as it appears after the cards have been key punched and listed on a printer. Likewise, the first few decision tables are shown on the coding sheets (photo reduced), the remainder are shown as they appear on the listing produced by the 7080 Decision Table System processor (also reduced). These two listings (data and decision tables) will constitute the documentation the user will receive when using the programming system.

The following list is an index to aid in reviewing the sample problem.

PG	LINE	NAME	CLASS	NUM.	DESCRIPTION	COMMENTS
			DATE		31262	
			MODEL	88		
			ASNBL		TABULAR	

IBM 7080

DECISION TABLE NAME DESCRIPTION SHEET

PROGRAM ______
DATE ______
PREPARED BY ______
PAGE ______
NEXT PAGE ______

PG	LINE	C	TYP	TABLE NO.	IDENT	TABLE NAME
00	0.0.0	0	A		TABLE	BEGIN

IDENT. 70X0T.P

PG	LINE	NAME	CLASS	NUM.	DESCRIPTION	COMMENTS
00			ENT80			
			LASN		@25000	
			SPL		@0000	
			NOSTP		B3¤T¤PROGHLT¤PROGERR¤¤	
			EEM			
			ASU	11¤		
			ASU	22¤		
			ASU	33¤		
			ASU	44¤		
			ASU	55¤		
			ASU	66¤		
			ASU	77¤		
			ASU	88¤		
			ASU	99¤		
			ASU	1010¤		
			ASU	1111¤		
			ASU	1212¤		
			ASU	1310¤		
			ASU	144¤		
			TR		TAB0005	

IBM 7080

DECISION TABLE NAME DESCRIPTION SHEET

PROGRAM ____________

DATE ____________

PREPARED BY ____________

PAGE ____________

NEXT PAGE ____________

PG	LINE	C C	TYP	TABLE NO.	IDENT	TABLE NAME	
01	0000	0	D	0000	TABLE	IODEFIN	

IDENT.

PG	LINE	NAME	CLASS	NUM.	DESCRIPTION	COMMENTS
01	01		IOCS			
	02		LASN		@500	
	03		IOTA		INMASTER⧧2000⧧2001⧧	
	04		IOTA		PAYREG⧧2002⧧2003⧧	
	05		IOTA		PAYCHECKS⧧2004⧧	
	06		IOTA		TERMCHECKS⧧2005⧧	
	07		IOTA		INTRANS⧧2100⧧2101⧧	
	08		IOTA		OUTMASTER⧧2102⧧2103⧧	
	09		IOTA		EXCEPTIONS⧧2104⧧	
	10		IOTA		MISCRPTS⧧2105⧧	
	11		IOTA		END⧧	
	12		LASN		@30000	
	13					
	14					
	15					
	16					
	17					
	18					
	19					
	20					
	21					
	22					
	23					
	24					
	25					

FORM X22-6913 PRINTED IN U. S. A.

7080 IOCS FILE TABLE MACRO-INSTRUCTIONS (LONG FORM)

PROGRAM: ______ CODED BY: ______ DATE: ______

03020 IOFTA PAYMASTERb □ 000 □ IORETURNTO □ IORETURNTO □ ~~IORETURNTO~~ MASTRINEOF □

03030
HI 0 RWD1 EORb STANDARD STANDARDb
HI □ 1 2 □ RWD2 □ EOFb □ NONSTAND □ TMbb □ ~~SPECSTANb~~ □ HSKRWD □
LO 3 RUN1 BOTH NOHEADER NOTM NONSTANDb NORWDb
LO 4 RUN2 NONE NOTRAILER

03040
PRIME SINGLE SEQUEN CKPTRCDSb DUMP ~~DELAYOPEN~~
NOPRI □ MULTIb □ NONSEQ □ NOCKPTRCD □ NODP □ ~~NODELAYbb~~ □

- 3.5 -

03050 IOFTB IP □ DATA □ 9 □ INITb □ CKLNG □ CMPCK □ 4 □ GENAREA □ F □ 150 □ 1500 □
OP WORK STACK NOCHK NOCMP AREAGIV V
PROG

03060
GETb MASTERINbb PAD BOTH BSDb
PUTb □ ~~NOWORKAREA~~ □ NOPA □ DELET □ WORK □
BOTH 00 EOF9S WBSD
NONE NONE NONE

IOFTC □ CKLNG □ CMPCK □ GP □ F □ □
NOCHK NOCMP NO V

IBM 7080

DECISION TABLE NAME DESCRIPTION SHEET

PROGRAM

DATE

PREPARED BY

PAGE

NEXT PAGE

PG	LINE	CC	TYP			IDENT	TABLE NAME	
	000	0				TABLE		

IDENT.

PG	LINE	NAME	CLASS	NUM.	DESCRIPTION	COMMENTS
05	010		FILE		TAPE 2000, 2001	150 CHAR MASTER RECORD FIXED INPUT
		MASTERIN	RNAME	A	MIEND	
		MIMANNO	FLD	5	N	MAN NUMBER
		MINAME		17	A+	MAN NAME
		MISTATUSCD		1	A	STATUS CODE, A-ACTIVE, T-TERMINATED
						I-INACTIVE
		MIORG		5	N	ORGANIZATION NUMBER
		MISHIFT		1	N	SHIFT EMPLOYEE WORKS
		MISOCSECNO		9	N	SOCIAL SECURITY NUMBER
		MISEX		1	A+	SEX OF EMPLOYEE
		MIBIRTHDAT		6	N	BIRTH DATE
		MIHIREDATE		6	N	HIRE IN DATE
		MINEXTVACD		6	N	NEXT VACATION ELIGIBILITY DATE
		MIDEDCODE	BITCD			DEDUCTION CODE
		MIINSDED		1		INSURANCE
		MILOAN		2		LOAN
		MIMISC		4		MISC DEDUCTIONS
			FLD	1	A+	BLANK
		MINODEPEND		2	+	NO OF DEPENDANTS
		MIRATE		4	#+01.03	RATE
		MIYTDHRS		6	#+05.01	YEAR-TO-DATE HOURS
		MIYTDMONEY		7	#+05.02	Y-T-D MONEY
		MIYTDPDWK		7	#+05.02	Y-T-D PAID WORKED TIME
		MIYTDPDUWK		7	#+05.02	Y-T-D PAID UNWORKED TIME
		MIQTDMONEY		7	#+05.02	QUARTER-TO-DATE MONEY

- 3.6 -

IBM 7080

DECISION TABLE NAME DESCRIPTION SHEET

PROGRAM ____
DATE ____
PREPARED BY ____
PAGE ____
NEXT PAGE ____

PG	LINE	C CP	TYP			IDENT	TABLE NAME	
	0 0 0	0				TABLE		

IDENT. 75 — 80

PG	LINE	NAME	CLASS	NUM.	DESCRIPTION	COMMENTS
0.6	0	MIYTDWTAX		6	#+04.02	Y-T-D WITHOLDING TAX
		MIYTDFICA		5	#+03.02	Y-T-D FICA
		MISLHRSUSD		5	#+04.01	SICK LEAVE HOURS USED
		MISLHRSACC		5	#+04.01	SICK LEAVE HOURS ACCUMULATED
		MIVACHRSUD		4	#+03.01	VACATION HOURS USED
		MIVACHRSAC		4	#+03.01	VACATION HOURS ACCUMULATED
		MIINSAMT		5	#+03.02	INSURANCE AMOUNT TO DEDUCT
		MILOANPAY		5	#+03.02	PAYMENT AMOUNT FOR LOAN
		MILOANBAL		6	#+04.02	BALANCE OF LOAN
		MIMDEDAMT		5	#+03.02	AMOUNT OF MISC DEDUCTIONS
		MIEND		1		TERMINAL RECORD MARK

```
08010             TITLE  FILE TABLE FOR OUTPUT MASTER
08020OUTMASTER IOFTA  PAYMASTER¤030¤IORETURNTO¤IORETURNTO¤IORETURNTO¤
08030                 HI ¤1¤RWD1¤NONE¤STANDARD¤TM  ¤STANDARD ¤HSKRWD¤
08040                 NOPRI¤SINGLE¤SEQUEN¤NOCKPTRCD¤NODP¤¤
08050             IOFTB  OP¤DATA¤9¤STACK¤CKLNG¤CMPCK¤2¤GENAREA¤F¤150¤1500¤
08060                 PUT ¤MASTEROUT¤PAD9¤NONE¤NONE¤
09010             FILE   TAPE 2102,2103  150 CHAR MASTER FIXED OUTPUT
09020MASTEROUT RNAME AMOEND
10010MOYTDWTAX        6(/04.02           Y-T-D WITHOLDING TAX
09030MOMANNO    FLD   5N                 MAN NUMBER
09040MONAME          17A/                MAN NAME
09050MOSTATUSCD       1A                 STATUS CODE,A-ACTIVE,T-TERMINATED
09060                                    I-INACTIVE
09070MOORG            5N                 ORGANIZATION NUMBER
09080MOSHIFT          1N                 SHIFT EMPLOYEE WORKS
09090MOSOCSECNO       9N                 SOCIAL SECURITY NO.
09100MOSEX            1A/                SEX
09110MOBIRTHDAT       6N                 BIRTH DATE
09120MOHIREDATE       6N                 HIRE DATE
09130MONEXTVACD       6N                 NEXT VACATION ELIGIBILITY DATE
09140MODEDCODE BITCD                     DEDUCTION CODE
09150MOINSDED                            INSURANCE
09160MOLOAN           2                  LOAN
09170MOMISC           4                  MISC
09180           FLD   1A/                BLANK
09190MONODEPEND       2/                 NO.OF DEPENDANTS
09200MORATE           4(/01.03           RATE
09210MOYTDHRS         6(/05.01           YEAR-TO-DATE HOURS
09220MOYTDMONEY       7(/05.02           YEAR-TO-DATE MONEY
09230MOYTDPDWK        7(/05.02           Y-T-D PAID WORK TIME
09240MOYTDPDUWK       7(/05.02           Y-T-D PAID UNWORKED TIME
09250MOQTDMONEY       7(/05.02           QUARTER TO DATE MONEY
10020MOYTDFICA        5(/03.02           Y-T-D FICA
10030MOSLHRSUSD       5(/04.01           SICK LEAVE HOURS USED
10040MOSLHRSACC       5(/04.01           SICK LEAVE HOURS ACCUMULATED
10050MOVACHRSUD       4(/03.01           VACATION HOURS USED
10060MOVACHRSAC       4(/03.01           VACATION HOURS ACCUMULATED

10070MOINSAMT         5(/03.02           AMOUNT FOR INSURANCE DEDUCTION
10080MOLOANPAY        5(/03.02           PAYMENT AMOUNT FOR LOAN
10090MOLOANBAL        6(/04.02           BALANCE OF LOAN
10100MOMDEDAMT        5(/03.02           AMOUNT OF MISC. DEDUCTIONS
10110MOEND      PRE   1
                TERMINAL RECORD MARK                 7080TP
```

```
12010          TITLE  FILE TABLE FOR INPUT TRANSACTIONS
12020INTRANS   IOFTA  TRANSACTIN□000□IORETURNTO□IORETURNTO□TRANSINEOF□
12030                 AI □1□RWD1□NONE□STANDARD□TM  □STANDARD □HSKPRWD□
12040                 PRIME□SINGLE□SEQUEN□NOCKPTRCD□NODP□□
12050          IOFTB  IP□DATA□8□STACK□CKLNG□CMPCK□2□GENAREA□F□80□800□
12060                 GET □TRANSIN□NONE□EOF9S□NONE□
13010          FILE   TAPES 2100,2101 80 CHAR FIXED TRANSACTION INPUT
13020TRANSIN   RNAME ATIEND
13030TRCD      FLD  02N                TRANSACTION CODE
13035TRMANNO         5N
13040               72A
13050TIEND           1
                                                  7080TP
```

```
14010          TITLE  FILE TABLE FOR PAY CHECKS RECORDS
14020PAYCHECKS IOFTA  REGPAYCHKS□010□IORETURNTO□IORETURNTO□IORETURNTO□
14030                 HI □0□RWD1□NONE□STANDARD□TM  □STANDARD □HSKRWD□
14040                 NOPRI□SINGLE□SEQUEN□NOCKPTRCD□NODP□□
14050          IOFTB  OP□DATA□6□STACK□CKLNG□CMPCK□2□GENAREA□F□095□0950□
14060                 PUT □REGCHECK□PAD □NONE□NONE□
14070          FILE   REGULAR PAY CHECKS- 95 CHARS-TAPE 2004
14080REGCHECK  RNAME ARPAYCKEND
14090          FLD  93A
14100RPAYCKEND       2
                                                  7080TP
```

```
15010          TITLE  FILE TABLE FOR TERMINATION CHECKS
15020TERMCHECKSIOFTA  TERMCHECKS□010□IORETURNTO□IORETURNTO□IORETURNTO□
15030                 HI □0□RWD1□NONE□STANDARD□TM  □STANDARD □HSKRWD□
15040                 NOPRI□SINGLE□SEQUEN□NOCKPTRCD□NODP□□
15050          IOFTB  OP□DATA□1□STACK□CKLNG□CMPCK□2□GENAREA□F□095□0950□
15060                 PUT □TERMCHECK□PAD □NONE□NONE□
15070          FILE   TERMINATION PAY CHECK - 95 CHAR RECORD - TAPE 2005
15080TERMCHECK RNAME ATCHECKEND
15090          FLD  93A
15100TCHECKEND       2
                                                  7080TP
```

```
16010          TITLE  EXCEPTION RECORD FILE
16020EXCEPTIONSIOFTA  PAYROLLEXP□010□IORETURNTO□IORETURNTO□IORETURNTO□
16030                 HI □0□RWD1□NONE□STANDARD□TM  □STANDARD .□HSKRWD□
16040                 NOPRI□SINGLE□SEQUEN□NOCKPTRCD□NODP□DELAYOPEN□
16050          IOFTB  OP□DATA□1□STACK□CKLNG□CMPCK□2□GENAREA□F□080□0800□
16060                 PUT □EXCEPTREC□PAD9□NONE□NONE□
16070          FILE   EXCEPTION RECORDS- 80 CHARS FIXED- TAPE 2104
16080EXCEPTREC RNAME AEXPRECEND
16090EXCEPTINFOGNAME  EXCEPTREA
16093EXTRCODE  FLD   2N                EXCEPTION TRANSACTION NUMBER
16095EXTRMANNO       5N                MANNO
16097EXCEPTREA      25A                EXCEPTION REASON
16099               47
16100EXPRECEND       1
```

```
                                                  7080TP
17010           TITLE  MISC REPORTS FILE
17020MISCRPTS   IOFTA  MISCREPORT□010□IORETURNTO□IORETURNTO□IORETURNTO□
17030                  HI □0□RWD1□NONE□STANDARD□TM  □STANDARD □HSKRWD□

17040                  NOPRI□SINGLE□SEQUEN□NOCKPTRCD□NODP□DELAYOPEN□
17050           IOFTB  OP□DATA□2□STACK□CKLNG□CMPCK□2□GENAREA□F□080□0800□
17060                  PUT □MISCREC□PAD9□NONE□NONE□
17070           FILE   MISC REPORT RECORDS - 80 CHARS FIXED -TAPE 2105
17080MISCREG    RNAME AMISCRECEND
17090           FLD  79
17100MISCRECEND        1
                                                  7080TP
18010           TITLE  PAYROLL REGISTER FILE-PREPARED BY REPORT WRITER
18020PAYREG     IOFTA  PAYROLLREG□010□IORETURNTO□XXXXAREOR□IORETURNTO□
18030                  HI □1□RWD1□NONE□STANDARD□TM  □STANDARD □HSKRWD□
18040                  NOPRI□SINGLE□SEQUEN□NOCKPTRCD□NODP□□
18050           IOFTB  OP□DATA□8□STACK□NOCHK□CMPCK□2□AREAGIV□F□125□1250□
18060           IOFTC  2□NOCHK□CMPCK□NO□F□PAYREGREC1□PAYREGREC2□
18070           FILE   PAYROLL REGISTER RECORD AREAS-USED BY REPORT WRITER
18080PAYREGREC1RNAME APREGR1END           TAPE 2002-2003
18090           FLD  50
18100PREGR1END      1200                  PAY REGISTER REC 1 END
18110           PRE    1                  GROUP MARK TO STOP WRITING AREA 1
18120           FILE   PAYROLL REGISTER RECORD AREA2 FOR REPORT WRITER
18130PAYREGREC2RNAME APREGR2END
18140           FLD  50
18150PREGR2END      1200                  PAY REGISTER REC 2 END
18160           PRE    1                  GROUP MARK TO STOP WRITING AREA 2
```

```
300100D     0001TABLE  CORE FILES
31010            FILE   HOLD AREA FOR TRANSACTION 00-CONTROL INFORMATION
31020TRCD00      RNAME ATR00END
31030TC00        FLD    2N                 TRANSACTION CODE 00
31040TCDUMMANNO         5N                 DUMMY MANNO OF ZEROS FOR SORT
31050TCCUTOFDATGNAME    TCCODA             PAY PERIOD CUTOFF DATE
31060TCCOYR      FLD    2N                 YEAR
31070TCCOMO             2N                 MONTH
31080TCCODA             2N                 DAY
31090TCPAYDATE GNAME    TCPDDA             PAY PERIOD PAY DATE
31100TCPDYR      FLD    2N                 YEAR
31110TCPDMO             2N                 MONTH
31120TCPDDA             2N                 DAY
31130                  60A
31140TR00END            1
                 RECORD MARK                              7080TP
32010            FILE   HOLD AREA FOR TRANSACTION 01 -NEW HIRES
32020TRCD01      RNAME ATR01END
32025TRCD1PART1GNAME    NHRATE             GNAME FOR SETUP OF NEW HIRE MASTER
32030TC01        FLD    2N
32040NHMANNO            5N
32050NHNAME            17A/
32060NHSTATUSCD         1A
32070NHORG              5N
32080NHSHIFT            1N
32090NHSOCSECNO         9N
32100NHSEX              1A/
32110NHBIRTHDAT         6N
32120NHHIREDATE         6N
32130NHNEXTVACD         6N
32140NHDEDCODE BITCD
32150NHINSDED           1
32160NHLOAN             2                  INVALID FOR A NEW HIRE
32170NHMISC             4
32180            FLD    1A/
32190NHNODEPEND         2/
32200NHRATE             4(/01.03
32210NHINSAMT           5(/03.02
32220NHMDEDAMT          5(/03.02
32230TR01END            3
```

```
                                                7080TP
33010           FILE   CREATE MASTER TRANSACTION 02,PART 1 OF A MASTER
33020TRCD02     RNAME ATR02END
33030TC02       FLD    2N
33031CMPART1    GNAME  CMYTDHRS
33040CMMANNO    FLD    5N
33050CMNAME           17A/
33060CMSTATUSCD        1A
33080CMORG             5N
33090CMSHIFT           1N
33100CMSOCSECNO        9N
33110CMSEX             1A/
33120CMBIRTHDAT        6N
33130CMHIREDATE        6N
33140CMNEXTVACD        6N
33150CMDEDCODE BITCD
33160CMINSDED          1
33170CMLOAN            2
33180CMMISC            4
33190           FLD    1A/
33200CMNODEREND        2/
33210CMRATE            4(/01.03
33220CMYTDHRS          6(/05.01
33230CMMDEDAMT         5(/03.02
33240TR02END           2
                                                7080TP
34010           FILE   CREATE MASTER-TRANSACTION  03,2ND PART OF A MASTER
34020TRCD03     RNAME ATR03END
34030TC03       FLD    2N
34040                  1A/              BLANK FOR LEFT PROTECTION
34050CMPART2    GNAME  CMLOANBAL
34060CMYTDMONEYFLD     7(/05.02
34070CMYTDPDWK         7(/05.02
34080CMYTDPDUWK        7(/05.02
34090CMQTDMONEY        7(/05.02
34100CMYTDWTAX         6(/04.02
34110CMYTDFICA         5(/03.02
34120CMSLHRSUSD        5(/04.01
34130CMSLHRSACC        5(/04.01
34140CMVACHRSUD        4(/03.01
34150CMVACHRSAC        4(/03.01
34160CMINSAMT          5(/03.02
34170CMLOANPAY         5(/03.02
34180CMLOANBAL         6(/04.02
34190TR03END           4
                                                7080TP
35010           FILE   TRANSACTION 04 - DELETE MASTER
35020TRCD04     RNAME ATR04END
35030TC04       FLD    2N
35040DMMANNO           5N
35050                 72
35060TR04END           1
                                                7080TP
```

```
36010          FILE   TRANSACTION 07 - WORK CARD, TIME WORKED,SICK LEAVE--
36020TRCD07    RNAME ATR07END
36030TC07      FLD   2N                TRANSACTION CODE OF 07
36040WCMANNO         5N                MAN NUMBER
36050WCORG           5N                ORGANIZATION
36060WCSHIFT         1N                SHIFT
36080WCPAYINLIU      1A                PAY-IN-LIEU CODE OF AN A
36090WCPAYADVAN      1A                ADVANCE PAY CHECK IF AN A
36100                1A/               BLANK
36110WCACTHRS        4(/03.01          ACTUAL HOURS WORKED
36120WCOTHRS         4(/03.01          OVERTIME HOURS WORKED
36130WCVACHRS        4(/03.01          VACTION HOURS TO BE PAID
36140WCSLHRS         4(/03.01          SICK LEAVE HRS TO BE PAID
36150WCHOLIHRS       4(/03.01          HOLIDAY HOURS TO BE PAID
36160WCPILHRS        4(/03.01          PAY-IN-LIEU HOURS
36170WCPADVHRS       4(/03.01          PAY-IN-ADVANCE HOURS
36180               35
36190TR07END         1
               RECORD MARK                        7080TP
37010          FILE   TRANSACTION 08 - TERMINATION RECORD
37020TRCD08    RNAME ATR08END
37030TC08      FLD   2N                TRANSACTION CODE 08 IS A TERMINATIO
37040TMMANNO         5N                MANNO
37050TMDATE          6N                DATE
37060TMSERVPAY       1A                SEVERANCE PAY CODE- A IS YES
37070                1A/               BLANK
37080TMSPAYAMT       5(/03.02          AMOUNT OF SEVERANCE PAY
37090               59
37100TR08END         1
```

```
38010            FILE   MASTER WORK AREA-FOR SETTING-UP NEW-MASTERS,ETC
38020WAMASTER    RNAME AWA01END
38025WAMASPART1GNAME   WAYTDHRS
38026WAMASNHPT1GNAME   WARATE           GNAME FOR SETUP OF NEW HIRE MASTER
38030WAMANNO     GNAME  WAMANNO4
38032WAMANNO1    FLD   01N
38034WAMANNO4           4N
38040WANAME            17A/
38050WASTATUSCD         1A
38060WAORG              5N
38070WASHIFT            1N
38080WASOCSECNO         9N
38090WASEX              1A/
38100WABIRTHDATGNAME   WABDDA
38110WABDYR      FLD    2N              BIRTH DATE YEAR
38120WABDMO             2N                         MONTH
38130WABDDA             2N                         DAY
38140WAHIREDATEGNAME   WAHDDA
38150WAHDYR      FLD    2N              HIRE DATE YEAR
38160WAHDMO             2N                        MONTH
38170WAHDDA             2N                        DAY
38180WANEXTVACDGNAME   WANVDA
38190WANVYR      FLD    2N              NEXT VACATION DATE YEAR
38200WANVMO             2N                                 MONTH
38210WANVDA             2N                                 DAY
38220WADEDCODE BITCD
38230WAINSDED           1
38240WALOAN             2
38250WAMISC             4
38260            FLD    1A/
38270WANODEPEND         2/
38270WARATE             4(/01.03
38279WAMASNHACCGNAME   WAMDEDAMT        GNAME FOR ZEROING OUT NHIRE ACC FLDS
38280WAYTDHRS    FLD    6(/05.01
39010WAMASPART2GNAME   WALOANBAL
39020WAYTDMONEYFLD      7(/05.02
39030WAYTDPDWK          7(/05.02
39040WAYTDPDUWK         7(/05.02
39050WAQTDMONEY         7(/05.02
39060WAYTDWTAX          6(/04.02
39070WAYTDFICA          5(/03.02
39080WASLHRSUSD         5(/04.01
39090WASLHRSACC         5(/04.01
39100WAVACHRSUD         4(/03.01
39110WAINSAMT           5(/03.02
39120WALOANPAY          5(/03.02
39130WALOANBAL          6(/04.02
39140WAMDEDAMT          5(/03.02
39150WA01END     PRE    1
```

7080TP

```
40010      FILE   PAY CHECK RECORD-BOTH REGULAR / TERMINATION
40020PAYCHECK  RNAME APCHECKEND
40030PCMANNO   FLD   5N
40040PCNAME          17A/
40050PCORG           5N
40060PCSHIFT         1N.
40070PCCUTOFFDA      6N
40080PCPAYDATE       6N
40090                1A/
40100PCHOURSWK       4(/03.01          HOURS WORKED IN THIS PAY PERIOD
40110PCHOURSPD       4(/03.01          HOURS PAID IN THIS PAY PERIOD
40120PCGROSS         7(/05.02          GROSS PAY FOR CURRENT PERIOD
40130PCNET           7(/05.02          NET FOR CURRENT PERIOD
40140PCWTAX          6(/04.02          WITHHOLDING TAX FOR CURRENT PERIOD
40150PCFICA          5(/03.02          CURRENT FICA
40160PCLOANDED       5(/03.02          CURRENT LOAN DEDUCTION
40170PCINSDED        5(/03.02          CURRENT INSURANCE DED
40180PCMISCDED       5(/03.02          CURRENT MISC DED
40185                5A
40190PCHECKEND PRE   1
```

7080TP

```
500100D    0002TABLE  VARIABLES
50020CUTOFFDATEGNAME  COYEAR            DATE FOR PAYROLL REGISTER
50030COMONTH   PRE   202
50040                1-
50050CODAY           212
50060                1-
50070COYEAR          262
50080EOFSWS    BITCD
50090EOFONMAST       1                  ON IF EOF HAS OCCURRED ON MASTER
50100EOFONTRANS      2                  ON IF EOF HAS OCCURRED ON TRANSACTNS
50110ORGDESCRIPFLD  15A                 DESCRIPTION OF ORGANIZATION
50120PAYDATE   GNAME  PDYEAR
50130PDMONTH   PRE   202
50140                1-
50150PDDAY           212
50160                1-

50170PDYEAR          262
50310          TITLE  AREAS FOR SEQUENCE CHECKING MASTERS AND TRANSACTIONS
50320PREVMASNO FLD   5N
50330PREVTRANNOFLD   5N
50340          TITLE  BITCD TO INDICATE RELATIONSHIP BETWEEN PREVIOUS
50350                 TRANSACTION AND PRESENT TRANSACTION
50360TRANSCOMP BITCD
50370PTHIGH          1                  PREVIOUS TRANS HIGH TO PRESENT
50380PTLOW           2                  PREVIOUS IS LOW TO PRESENT
50390PTEQUAL         4                  PREVIOUS IS EQUAL TO PRESENT
50400DUPMASTER GNAME ADUPMASEND
50410          PRE  22DUPLICATE MAN NUMBER
50420DUPMANNO        512346
50430DUPMASEND       3
50440MASSEQERR GNAME AMASSEQERR1
50450          PRE   7MASTER
50460SEQERMANNO      512346
50470               17 OUT OF SEQUENCE
50480MASSEQERR1      1
50600NHCONZERO GNAME  NHCONZERON        ZEROES FOR NEW HIRE MASTER
50610          PRE  5200000/000000/000000/000000/000000/00000/0000/0000/00
50620NHCONZERON     2800/000/0000/0000/00000/0000/
50630          TITLE  MATH WORK AREAS
50640          PRE  01
50650FICAWORK  FLD  051/03.02           FICA WORK AREA
50660DEDWORK   FLD  061/04.02           DEDUCTION WORK AREA
51010          TITLE  REASONS FOR A TRANSACTION BEING AN EXCEPTION
51020EXREA1    PRE  25NO MASTER RECRD
51030EXREA2         25TRANS FOR NH NOT 07
51040EXREA3         25TRANS FOR DELETED MASTER
51050EXREA4         25TRANS 02 WITHOUT AN 03
51100EXREA5         25INVALID ORG FOR NEW HIRE
51110EXREA6         25TWO TRANS 7 FOR 1 MASTER
51120EXREA7         25INSUFF NET FOR DEDUCTION
```

```
550100D     0003TABLE  EXPRESSIONS
55020            TITLE  THE FOLLOWING VALUE LIST IS USED TO FIND THE
55030                   ENGINEERING ORGS THAT RECEIVE A BONUS
55040ORGVLIST    VLIST 510000
55050                   11010
55060                   43200
55070                   45000
55080                   71000
55090                   82300
55100            TITLE  THE FOLLOWING REFERENCE LIST IS USED TO CONVERT
55110                   ORGANIZATION NUMBER TO A DESCRIPTION FOR A REPORT
55120ORGRLIST    RLIST  05,15,ORGDESCRIP
55130                   10000,ENG. GROUP 1 A
55140                   10500,MANUFACTING 3 A
55150                   11010,ENG. SCHEDULING
55160                   25000,QUALITY CONTROL
55170                   33000,PRODUCTION CTL
55180                   43200,ENG. RESEARCH
55190                   45000,ENG. EXPERIMENT
55200                   51000,PLANT SERVICES
55210                   61000,PLANT MAINT.
55220                   71000,ENG. PLANNING
55230                   82300,ENG. DRAWING
55240                   85000,PLANT BUDGET
55250                   91000,DATA PROCESSING
56010            TITLE  CALCULATE TOTAL HOURS TO BE PAID
56020TOTALHRSPDMEXP     (/03.01 WCOTHRS * 1.5 / WCACTHRS / WCVACHRS /
56030                   WCSLHRS / WCHOLIHRS
56040            TITLE  CALCULATE TOTALMONEY - HOURS TIMES RATE
56050TOTALMONEYMEXP     (/05.02 PCHOURSPD * WARATE
56060            TITLE  TOTAL HOURS WORKED IS OVERTIME HRS PLUS ACTUAL HRS
56070TOTALHRSWKMEXP     (/03.01 WCOTHRS / WCACTHRS
56080            TITLE  NET PAY IS GROSS MINUS WTAX MINUS FICA
56090CALNETPAY   MEXP   (/05.02 PCGROSS - PCWTAX - PCFICA
56100            TITLE  CALCULATE WITHOLDING TAX
56110CALWTAX     MEXP   (/04.02 % PCGROSS - WANODEPEND * 13.00 □ * .18
56120            TITLE  COMPUTE FICA
56130CALFICA     MEXP   (/03.02 PCGROSS * .03125
56140            TITLE  COMPUTE BONUS FOR ENGINEERING DEPARTMENTS
56150ENGBONUS    MEXP   (/05.02 PCGROSS * .03
57010            TITLE  JOIN TRANS NO., MANNO AND EXCEPTION REASON
57020                   FOR EXCEPTION MESSAGE RECORD
57030SEXREXREA1SEXP 32JOIN%TRCD,TRMANNO,EXREA1□
57040SEXREXREA2SEXP 32JOIN%TRCD,TRMANNO,EXREA2□
57050SEXREXREA3SEXP 32JOIN%TRCD,TRMANNO,EXREA3□
57060SEXREXREA4SEXP 32JOIN%TRCD,TRMANNO,EXREA4□
57070SEXREXREA5SEXP 32JOIN%TC01,NHMANNO,EXREA5□
57080SEXREXREA6SEXP 32JOIN%TRCD,TRMANNO,EXREA6□
57090SEXREXREA7SEXP 32JOIN%TC07,WCMANNO,EXREA7□
```

IBM 7080

DECISION TABLE LOGIC SHEET

PROGRAM IDENTIFICATION

PAGE 60 | LINE 0000 | L01 | TABLE NO. 0005 | IDENTIFICATION TABLE | TABLE NAME HOUSEKEEP1 | NEXT TABLE NAME TAB0006 | ERROR TABLE NAME

ORDER OF CONDITION ROWS: LINE 0001 | IDENTIFICATION CONDORDER

ORDER OF RULES: LINE 0002 | IDENTIFICATION RULEORDER

PROGRAM ______ DATE ______

PREPARED BY ______ PAGE ______

COMMENT: PAGE 60 | LINE 0003 | IDENTIFICATION TITLE | PART 1 OF HOUSEKEEPING

NEXT PAGE ______

START NEXT CARD: Dup. Col's. 1–5, Punch B in Col. 6, Skip Col's. 7–23, Dup. Col's. 75–80.

PAGE	LINE	C	ACTION OPERATOR	NAME 1	OP	NAME 2	RULE 1	RULE 2	RULE 3	RULE 4	RULE 5	RULE 6
	01	A										
60	02	A	OPEN				X					
	03	A	MOVE	'X'		TO TRANSCOMP	X					
	04	A	MOVE	'X'		TO MASERRCODE	X					
	05	A	MOVE	BLANK		TO PREVMASNO	X					
	06	A	MOVE	BLANK		TO PREV. TRANNO	X					
	07	A	MOVE	'X'		TO EOFSWS	X					
	08	A	DO	GET TRANS			X					
	09	A	DO	GET MASTER			X					
	10	A										

PAGE 61 | LINE 0000 | 0 | TABLE NO. 020006 | IDENTIFICATION TABLE | TABLE NAME HOUSEKEEP2 | NEXT TABLE NAME | ERROR TABLE NAME TABSTOP

ORDER OF CONDITION ROWS: LINE 0001 | IDENTIFICATION CONDORDER

ORDER OF RULES: LINE 0002 | IDENTIFICATION RULEORDER

PROGRAM ______ DATE ______

PREPARED BY ______ PAGE ______

COMMENT: PAGE 61 | LINE 0003 | IDENTIFICATION TITLE | PART 2 OF HOUSEKEEPING

NEXT PAGE ______

START NEXT CARD: Dup. Col's. 1–5, Punch B in Col. 6, Skip Col's. 7–23, Dup. Col's. 75–80.

PAGE	LINE	C	ACTION OPERATOR	NAME 1	OP	NAME 2	RULE 1	RULE 2	RULE 3	RULE 4	RULE 5	RULE 6
61	01	A		TRCD	EQ	00	Y	N				
	02	A										
	03	A	MOVE	TRANSIN		TO TRCD00	X					
	04	A	MOVE	TCCOYR		TO COYEAR	X					
	05	A	MOVE	TCCOMO		TO COMONTH	X					
	06	A	MOVE	TCCODA		TO CODAY	X					
	07	A	MOVE	TCPDYR		TO PDYEAR	X					
	08	A	MOVE	TCTDMO		TO PDMONTH	X					
	09	A	MOVE	TCTDDA		TO PDDAY	X					
	10	A	DO	GETTRANS			X					
	11	A	GO	TO TAB0009			X					
	12	A	GO	TO TABSTOP				X				
	13	A										

3

PROG DATE TABLE TAB0009 ELSE RULE 08 NEXT TABLE ERROR TABLE TAB0013 PAGE NO 1

COMMENTS DETERMINE TRANS TYPE AND PROCESS A

PG LN	ACTION	NAME 1	OP NAME 2		01	02	03	04	05	06	07	08	CON NO
6301 A		TRMANNO	VS MIMANNO	•	GR	EQ	LR	LR	LR	EQ	EQ		01
6302 A		TRCD	EQ 01	•			Y	N	N				02
6303 A		TRCD	EQ 02	•				Y	N				03
6304 A		TRCD	EQ 04	•		Y							04
6307 A		TRCD	EQ 07	•						Y			05
6308 A		TRCD	EQ 08	•							Y		06
6309 A	DO	GETMASTER		•	X								
6310 A	GOTO	TAB0009		•	X								
6311 A	MOVE	TRANSIN	TO TRCD01	•			X						
6312 A	MOVE	TRCD1PART1	TO WAMASNHPT1	•			X						
6313 A	MOVE	NHCONZERO	TO WAMASNHACC	•			X						
6315 A	GOTO	TAB0010		•			X						
6316 A	MOVE	TRANSIN	TO MISCREC	•		X							
6317 A	PUT	MISCRPTS		•		X							
6318 A	DO	GETMASTER		•		X							
6319 A	DO	GETTRANS		•		X							
6320 A	GOTO	TAB0025		•		X							
6321 A	MOVE	TRANSIN	TO TRCD02	•				X					
6322 A	DO	GETRANS		•				X					
6323 A	GOTO	TAB0030		•				X					
6324 A	SET	EXCEPTINFO	EQ SEXPEXREA1	•					X				
6325 A	PUT	EXCEPTIONS		•					X			X	
6326 A	GOTO	TAB0009		•					X			X	
6330 A	MOVE	MASTERIN	TO WAMASTER	•						X	X		

3

PROG DATE TABLE TAB0009 ELSE RULE 08 NEXT TABLE ERROR TABLE TAB0013 PAGE NO 2

COMMENTS DETERMINE TRANS TYPE AND PROCESS A

PG LN	ACTION	NAME 1	OP NAME 2		01	02	03	04	05	06	07	08	CON NO
6331 A	MOVE	TRANSIN	TO TRCD07	•						X			
6332 A	DO	GETTRANS		•						X			
6333 A	GOTO	TAB0060		•						X			
6334 A	MOVE	TRANSIN	TO TRCD08	•							X		
6336 A	GOTO	TAB0050		•							X		

PROG DATE TABLE MASTRINEOF TAB ELSE RULE NEXT TABLE DO ERROR TABLE PAGE NO 1

COMMENTS A

PG LN	ACTION	NAME 1	OP NAME 2		01	CON NO
6401 A	SET	EOFONMAST	EQ ON	•	X	
6402 A	GOTO	IORETURNTO		•	X	

PROG DATE TABLE TRANSINEOF TAB ELSE RULE NEXT TABLE DO ERROR TABLE PAGE NO 1

COMMENTS A

PG LN	ACTION	NAME 1	OP NAME 2		01	CON NO
6501 A	SET	EOFONTRANS	EQ ON	•	X	
6502 A	GOTO	IORETURNTO		•	X	

PROG DATE TABLE TAB0010 ELSE RULE NEXT TABLE ERROR TABLE PAGE NO 1

COMMENTS CHECK NEW HIRE FOR VALID ORG NUMBER A

PG LN	ACTION	NAME 1	OP NAME 2		01	02	CON NO
6601 A		NHORG	EQ ORGVLIST	•	Y	N	01
6602 A	SET	EXCEPTINFO	EQ SEXPEXREAS	•	X		
6603 A	PUT	EXCEPTIONS		•	X		
6604 A	DO	GETTRANS		•	X	X	
6605 A	GOTO	TAB0020		•	X	X	

PAGE	LINE	C C	TYPE	NO. RULES	ELSE	TABLE NO.	IDENTIFICATION	TABLE NAME	NEXT TABLE NAME	ERROR TABLE NAME
68	000	0	L	03	3	0020	TABLE			

PAGE	LINE	C C	IDENTIFICATION	ORDER OF CONDITION ROWS
	0001		CONDORDER	

PAGE	LINE	C C	IDENTIFICATION	ORDER OF RULES
	0002		RULEORDER	

PROGRAM ______ DATE ______

PREPARED BY ______ PAGE ______

PAGE	LINE	C C	IDENTIFICATION	COMMENT
68	0003		TITLE	PROCESS TRANS 7 OR 8 AFTER SETUP OF NEW HIRE MASTER

NEXT PAGE ______

START NEXT CARD: Dup. Col's. 1–5, Punch 8 in Col. 6, Skip Col's. 7–33, Dup. Col's. 75–80.

PAGE	LINE	C C	ACTION OPERATOR	NAME 1	OP	NAME 2	RULE 1 01	02	03	04	05	06	RULE 2 07–12	RULE 3 13–18	RULE 4 19–24	RULE 5 25–30	RULE 6 31–32
68	01	A		PTEQUAL	IS	ON	N	Y									
	02	A		TRCD	EQ	07		Y									
	03	A															
	04	A	MOVE	WAMASTER	TO	MASTEROUT	X		X								
	05	A	PUT	OUTMASTER			X		X								
	06	A	MOVE	TRCD01	TO	MISCREC	X		X								
	07	A	PUT	MISCRPTS			X		X								
	08	A	SET	EXCEPTINFO	EQ	SEXPEXREA2			X								
	09	A	PUT	EXCEPTIONS					X								
	10	A	MOVE	TRANSIN	TO	TRCD07		X									
	11	A	DO	GETTRANS				X	X								
	12	A	GO TO	TAB0009			X		X								
	13	A	GO TO	TAB1060				X									
	14	A															

```
PROG        DATE        TABLE           TAB0025  ELSE RULE    NEXT TABLE        ERROR TABLE        PAGE NO  1
COMMENTS    DELETED MASTER ROUTINE                                                                          A
- - - - - - - -  S T U B  - - - - - - - - - 01 02 03                                                      CON
PG LN  ACTION   NAME 1    OP   NAME 2                                                                    NO
7001 A          PTLOW      IS ON       * Y  N  N                                                          01
7002 A          PTEQUAL    IS ON       *    Y  N                                                          02
*********************************************************************************************************
7003 A  GOTO    TAB0009                * X
7004 A  SET     EXCEPTINFO EQ SEXPEXREA3  *    X
7005 A  PUT     EXCEPTIONS             *    X
7006 A  DO      GETTRANS               *    X  X
7007 A  GOTO    TAB0025                *    X  X
```

```
PROG        DATE        TABLE           TAB0030  ELSE RULE    NEXT TABLE        ERROR TABLE        PAGE NO  1
COMMENTS    CREATE MASTER FROM TRANSACTIONS 2 AND 3                                                         A
- - - - - - - -  S T U B  - - - - - - - - - 01 02 03                                                      CON
PG LN  ACTION   NAME 1    OP   NAME 2                                                                    NO
7301 A          PTEQUAL    IS OFF      * Y  N  N                                                          01
7302 A          TRCD       EQ 03       *    N  Y                                                          02
*********************************************************************************************************
7303 A  SET     EXCEPTINFO EQ SEXPEXREA4  * X  X
7304 A  PUT     EXCEPTIONS             * X  X
7305 A  GOTO    TAB0009                * X  X
7306 A  MOVE    TRANSIN    TO TRCD03   *       X
7307 A  MOVE    CHPART1    TO WAMASPART1  *    X
7308 A  MOVE    CMPART2    TO WAMASPART2  *    X
7309 A  MOVE    CMMDEDAMT  TO WAMDEDAMT   *    X
7311 A  DO      GETTRANS               *       X
7312 A  GOTO    TAB0031                *       X
```

```
PROG         DATE          TABLE              TAB0031  ELSE RULE    NEXT TABLE          ERROR TABLE             PAGE NO  1
COMMENTS     PROCESS TRANS AFTER CREATING NEW MASTER                                                                   A
- - - - - - - - - STUB - - - - - - - - - 01 02 03                                                                      CON
PG LN  ACTION  NAME 1     OP   NAME 2                                                                                  NO
7501 A         PTEQUAL    IS ON           • Y  Y  N                                                                    01
7502 A         TRCD       EQ 07           • Y  N                                                                       02
7503 A         TRCD       EQ 08           • N  Y                                                                       03
•••••••••••••••••••••••••••••••••••••••••••••••••••••••••••••••••••••••••••••••••••••••••••••••••••••••••••••••••••••••••
7504 A  MOVE   TRANSIN    TO TRCD07       • X
7505 A  DO     GETTRANS                   • X
7506 A  GOTO   TAB0060                    • X
7507 A  MOVE   TRANSIN    TO TRCD08       •    X
7508 A  GOTO   TAB0050                    •    X
7509 A  MOVE   WAMASTER   TO MASTEROUT    •       X
7510 A  PUT    OUTMASTER                  •       X
7511 A  GOTO   TAB0009                    •       X
```

```
PROG         DATE          TABLE              TAB0050  ELSE RULE    NEXT TABLE          ERROR TABLE             PAGE NO  1
COMMENTS     TERMINATION ROUTINE WITHOUT A WORK CARD - TR 07                                                           A
- - - - - - - - - STUB - - - - - - - - - 01 02 03                                                                      CON
PG LN  ACTION  NAME 1     OP   NAME 2                                                                                  NO
7601 A         TMSERVPAY  EQ (A(          • Y  Y  N                                                                    01
76012A         WAORG      EQ ORGRLIST     • Y  N                                                                       02
•••••••••••••••••••••••••••••••••••••••••••••••••••••••••••••••••••••••••••••••••••••••••••••••••••••••••••••••••••••••••
76013A  MOVE   BLANK      TO ORGDESCRIP   •    X
7602 A  MOVE   TMMANNO    TO PCMANNO      • X  X
7603 A  MOVE   WANAME     TO PCNAME       • X  X
7604 A  MOVE   WAORG      TO PCORG        • X  X
7605 A  MOVE   WASHIFT    TO PCSHIFT      • X  X
7606 A  MOVE   TMDATE     TO PCCUTOFFDA   • X  X
7607 A  MOVE   TMDATE     TO PCPAYDATE    • X  X
7608 A  SET    PCHOURSWK  EQ ZERO         • X  X
7609 A  SET    PCHOURSPD  EQ ZERO         • X  X
7610 A  MOVE   TMSPAYAMT  TO PCGROSS      • X  X
7611 A  SET    PCLOANDED  EQ ZERO         • X  X
7612 A  SET    PCINSDED   EQ ZERO         • X  X
7613 A  SET    PCMISCDED  EQ ZERO         • X  X
7614 A  SET    PCWTAX     EQ CALWTAX      • X  X
7615 A  SET    PCFICA     EQ CALFICA      • X  X
7616 A  MOVE   PCFICA     TO FICAWORK     • X  X
7617 A  SET    FICAWORK   /  WAYTDFICA    • X  X
7618 A  GOTO   TAB0051                    • X  X  X
```

```
PROG         DATE          TABLE              TAB0051  ELSE RULE    NEXT TABLE TAB0009  ERROR TABLE             PAGE NO  1
COMMENTS     CONTINUE PROCESSING TERMINATION                                                                           A
- - - - - - - - - STUB - - - - - - - - - 01 02 03                                                                      CON
PG LN  ACTION  NAME 1      OP   NAME 2                                                                                 NO
7701 A         TMSERVPAY   EQ (A(          • Y  Y  N                                                                   01
7702 A         FICAWORK    GR 150.00       • N  Y                                                                      02
•••••••••••••••••••••••••••••••••••••••••••••••••••••••••••••••••••••••••••••••••••••••••••••••••••••••••••••••••••••••••
7703 A  SET    FICAWORK    -  150.00       •    X
7704 A  SET    PCFICA      -  FICAWORK     •    X
7705 A  SET    WASTATUSCD  EQ (I(          • X  X  X
7706 A  SET    PCNET       EQ CALNETPAY    • X  X
7707 A  SET    WAYTDMONEY  /  PCGROSS      • X  X
7708 A  SET    WAYTDPDUWK  /  PCGROSS      • X  X
7709 A  SET    WAQTDMONEY  /  PCGROSS      • X  X
7710 A  SET    WAYTDWTAX   /  PCWTAX       • X  X
7711 A  SET    WAYTDFICA   /  PCFICA       • X  X
7712 A  MOVE   PAYCHECK    TO TERMCHECK    • X  X
7713 A  PUT    TERMCHECKS                  • X  X
7714 A  MOVE   TRCD08      TO MISCREC      • X  X  X
77141A  PUT    MISCRPTS                    • X  X  X
7715 A  DO     PUTPAYREG                   • X  X
7716 A  MOVE   WAMASTER    TO MASTEROUT    • X  X  X
7717 A  PUT    OUTMASTER                   • X  X  X
7718 A  DO     GETMASTER                   • X  X  X
7719 A  DO     GETTRANS                    • X  X  X
```

```
PROG       DATE        TABLE        TAB0060  ELSE RULE 07  NEXT TABLE TAB0061   ERROR TABLE   PAGE NO  1
COMMENTS   EDIT TRANSACTION NO. 7                                                              A
----------- S T U B ---------------------- 02 04 03 06 05 01 07                                CON
PG LN  ACTION NAME 1     OP NAME 2                                                             NO
7805 A       WACACHRS   IS POSITIVE     • Y  Y  Y  Y  Y                                        09
7806 A       WCOTHRS    IS POS          • Y  Y  Y  Y  Y                                        10
7807 A       WCVACHRS   IS POS          • Y  Y  Y  Y  Y                                        11
7808 A       WCSLHRS    IS POS          • Y  Y  Y  Y  Y                                        01
7809 A       WCHOLIHRS  IS POS          • Y  Y  Y  Y  Y                                        03
7810 A       WCPILHRS   IS POS          • Y  Y  Y  Y  Y                                        04
7811 A       WCPADVHRS  IS POS          • Y  Y  Y  Y  Y                                        02
7801 A       PTEQUAL    IS ON           • N  N  N  Y  Y  Y                                     05
7803 A       WCPAYINLIU EQ (A(          • N  N  Y  Y  N                                        07
7804 A       WCPAYADVAN EQ (A(          • N  Y  N  N  Y                                        08
7802 A       TRCD       EQ 08           •          Y  Y  N                                     06
•••••••••••••••••••••••••••••••••••••••••••••••••••••••••••••••••••••••••••••••••••••••••••••••••
7812 A MOVE  WCPILHRS   TO PCHOURSPD    •       X
7813 A MOVE  WCPADVHRS  TO PCHOURSPD    •    X
7814 A MOVE  ZERO       TO PCHOURSPD    • X        X  X
7815 A SET   PCHOURSPD  /  TOTALHRSPD   • X  X  X  X  X
7816 A SET   PCHOURSWK  EQ TOTALHRSWK   • X  X  X  X  X
7817 A SET   PCGROSS    EQ TOTALMONEY   • X  X  X  X  X
7818 A SET   PCNET      EQ CALNETPAY    • X  X  X  X  X
7819 A SET   PCWTAX     EQ CALWTAX      • X  X  X  X  X
7820 A SET   PCFICA     EQ CALFICA      • X  X  X  X  X
7821 A MOVE  PCFICA     TO FICAWORK     • X  X  X  X  X
7822 A SET   EXCEPTINFO EQ SEXPEXREA6   •                X  X
7823 A PUT   EXCEPTIONS                 •                X  X
7824 A DO    GETTRANS                   •                X  X

7825 A GOTO  TAB0060                    •                X  X
```

3

```
PROG       DATE        TABLE        TAB0061  ELSE RULE     NEXT TABLE TAB0062   ERROR TABLE   PAGE NO  1
COMMENTS   CONTINUE TRANS 7 PROCESS                                                            A
----------- S T U B ---------------------- 01 02 03 04                                         CON
PG LN  ACTION NAME 1     OP NAME 2                                                             NO
7903 A       FICAWORK   GR 150.00       • Y  N  Y  N                                           01
7904 A       WCORG      EQ ORGKLIST     • Y  N  N  Y                                           02
•••••••••••••••••••••••••••••••••••••••••••••••••••••••••••••••••••••••••••••••••••••••••••••••••
7905 A SET   FICAWORK   -  150.00       • X     X
7906 A SET   PCFICA     -  FICAWORK     • X     X
7907 A MOVE  WCMANNO    TO PCMANNO      • X  X  X  X
7908 A MOVE  WANAME     TO PCNAME       • X  X  X  X
7909 A MOVE  WCORG      TO PCORG        • X  X  X  X
7910 A MOVE  WCSHIFT    TO PCSHIFT      • X  X  X  X
7911 A MOVE  TCCUTOFDAT TO PCCUTOFFDA   • X  X  X  X
7912 A MOVE  TCPAYDATE  TO PCPAYDATE    • X  X  X  X
7913 A MOVE  WAINSAMT   TO PCINSDED     • X  X  X  X
7914 A MOVE  WALOANPAY  TO PCLOANDED-   • X  X  X  X
7915 A MOVE  WAMDEDAMT  TO PCMISCDED    • X  X  X  X
7917 A SET   WAYTDMONEY /  PCGROSS      • X  X  X  X
7918 A SET   WAYTDPDWK  /  PCGROSS      • X  X  X  X
7919 A SET   WAQTDMONEY /  PCGROSS      • X  X  X  X
7920 A SET   WAYTDFICA  /  PCFICA       • X  X  X  X
7921 A SET   WAYTDWTAX  /  PCWTAX       • X  X  X  X
7922 A MOVE  BLANK      TO ORGDESCRIP   •    X  X
7923 A MOVE  WAINSAMT   TO DEDWORK      • X  X  X  X
7924 A SET   DEDWORK    /  WALOANPAY    • X  X  X  X
7925 A SET   DEDWORK    /  WAMDEDAMT    • X  X  X  X
```

```
PROG       DATE        TABLE        TAB0062  ELSE RULE     NEXT TABLE           ERROR TABLE   PAGE NO  1
COMMENTS   CONTINUE TRANS 7 PROCESS                                                            A
----------- S T U B ---------------------- 01 02 03 04                                         CON
PG LN  ACTION NAME 1     OP NAME 2                                                             NO
8001 A       PTEQUAL    IS ON           • Y  Y  N  N                                           01
8002 A       TRCD       EQ 08           • Y  Y                                                 02
8003 A       DEDWORK    LR PCNET        • Y  N  Y  N                                           03
•••••••••••••••••••••••••••••••••••••••••••••••••••••••••••••••••••••••••••••••••••••••••••••••••
8004 A SET   PCNET      -  DEDWORK      • X     X
8005 A SET   EXCEPTINFO EQ SEXPEXREAT   •    X     X
8006 A PUT   EXCEPTIONS                 •    X     X
8007 A MOVE  PAYCHECK   TO REGCHECK     • X  X  X  X
8008 A PUT   PAYCHECKS                  • X  X  X  X
8009 A DO    PUTPAYREG                  • X  X  X  X
8010 A MOVE  TRANSIN    TO TRCD08       • X  X
8011 A GOTO  TAB0050                    • X  X
8012 A MOVE  WAMASTER   TO MASTEROUT    •       X  X
8013 A PUT   OUTMASTER                  •       X  X
8014 A DO    GETMASTER                  •       X  X
8015 A DO    GETTRANS                   •       X  X
8016 A GOTO  TAB0009                    •       X  X
```

```
PROG          DATE           TABLE EOFMROUT     TAB        ELSE RULE      NEXT TABLE                 ERROR TABLE TABSTOP      PAGE NO  1
COMMENTS      EOF ROUTINE FOR MASTER-GOTO EOJ-TRANS IS ALSO AT EOF                                                                 A
- - - - - - - - -  S T U B  - - - - - - - - - 01                                                                                  CON
PG LN  ACTION   NAME 1    OP    NAME 2                                                                                            NO
8101 A          EOFONTRANS IS ON           * Y                                                                                    01
*******************************************************************************************************************************
8102 A  GOTO  TAB0070                      * X
```

```
PROG          DATE           TABLE EOFTROUT     TAB0070  ELSE RULE      NEXT TABLE                 ERROR TABLE              PAGE NO  1
COMMENTS      EOJ ROUTINE-ENTERED WHEN EOF ON TRANS                                                                                A
- - - - - - - - -  S T U B  - - - - - - - - - 01                                                                                  CON
PG LN  ACTION   NAME 1    OP    NAME 2                                                                                            NO
8201 A  CLOSE  INMASTER                    * X
8202 A  CLOSE  PAYCHECKS                   * X
8203 A  CLOSE  TERMCHECKS                  * X
8204 A  CLOSE  INTRANS                     * X
8205 A  CLOSE  OUTMASTER                   * X
8206 A  CLOSE  EXCEPTIONS                  * X
8207 A  CLOSE  MISCRPTS                    * X
8208 A  GOTO   PROGHLT                     * X
```

```
PROG          DATE           TABLE GETTRANS     TAB        ELSE RULE      NEXT TABLE DO              ERROR TABLE              PAGE NO  1
COMMENTS      ROUTINE TO GET A TRANSACTION                                                                                         A
- - - - - - - - -  S T U B  - - - - - - - - - 01                                                                                  CON
PG LN  ACTION   NAME 1    OP    NAME 2                                                                                            NO
8501 A  GET    INTRANS                     * X
8502 A  DO     SEQTRANS                    * X
```

```
PROG          DATE           TABLE SEQTRANS     TAB        ELSE RULE      NEXT TABLE DO              ERROR TABLE              PAGE NO  1
COMMENTS      SEQUENCE CHECK THE TRANSACTIONS                                                                                      A
- - - - - - - - -  S T U B  - - - - - - - - - R U L E  01 .R U L E  02  R U L E  03  R U L E        R U L E        R U L E     CON
PG LN  ACTION   NAME 1    OP    NAME 2      OP    NAME 2  OP    NAME 2  OP    NAME 2  OP    NAME 2  OP    NAME 2  OP    NAME 2  NO
8601 A          PREVTRANNO VS TRMANNO      * LR            EQ            GR                                                      01
*******************************************************************************************************************************
8602 A  SET    PTLOW      EQ               *     ON            OF            OF
8603 A  SET    PTEQUAL    EQ               *     OF            ON            OF
8605 A  MOVE   TRMANNO    TO PREVTRANNO    * X
8604 A  SET    PTHIGH     EQ               *     OF            OF            ON
```

```
PROG          DATE           TABLE GETMASTER    TAB        ELSE RULE      NEXT TABLE DO              ERROR TABLE              PAGE NO  1
COMMENTS      GET A MASTER RECORD                                                                                                  A
- - - - - - - - -  S T U B  - - - - - - - - - 01                                                                                  CON
PG LN  ACTION   NAME 1    OP    NAME 2                                                                                            NO
8701 A  GET    INMASTER                    * X
8702 A  DO     SEQMASTER                   * X
```

```
PROG          DATE           TABLE SEQMASTER    TAB        ELSE RULE      NEXT TABLE DO              ERROR TABLE              PAGE NO  1
COMMENTS      SEQUENCE CHECK THE MASTER RECORDS                                                                                    A
- - - - - - - - -  S T U B  - - - - - - - - - R U L E  01  R U L E  02  R U L E  03  R U L E        R U L E        R U L E     CON
PG LN  ACTION   NAME 1    OP    NAME 2      OP    NAME 2  OP    NAME 2  OP    NAME 2  OP    NAME 2  OP    NAME 2  OP    NAME 2  NO
8801 A          PREVMASNO  VS MIMANNO      * LR            EQ            GR                                                      01
*******************************************************************************************************************************
8802 A  MOVE   MIMANNO    TO               *     PREVMASNO     DUPMANNO      SEQERMANNO
8803 A  TYPE   MASSEQERR                   *                             X
```

```
900100R    9800TABLEPUTPAYREG
90020           MODE   REPORT
90030PUTPAYREG DREPT  PAYREG,PAYREGREC1,PAYREGREC2,X10,P,54,120,EOR,
90040           PAHDG
90050           CARRC  1
90060           CONST42
90070           CONST33P A Y R O L L   R E G I S T E R
90080           PAHDG
90090           CARRC
90100           CONST39
90110           CONST12CUTOFF DATE
90120           WDATE08CUTOFFDATE,
90130           CONST11  PAY DATE
90140           WDATE08PAYDATE,
90150           PAHDG
90160           CARRC  0
90170           CONST52MANNO        NAME         SOC SEC NO   ORG       DESC
90180           CONST52          SH  VACDATE  NO.DEP  RATE   YTDHRS   YTDMONE
90190           CONST10Y  YTDWTAX
90200           PAHDG
90210           CARRC
90220           CONST52YTD FICA  SL ACC  VAC ACC  INS AMT  LOAN BAL  MISDED
90230           CONST52   CURHRSWK CURHRSPD CURGROSS CURNET   CURWTAX  CURF
90240           CONST04ICA
91010           DTAIL
91020           CARRC
91030           RECRD05WAMANNO,
91040           CONST02
91050           RECRD17WANAME,
91060           CONST02
91070           RECRD09WASOCSECNO,
91080           CONST02
91090           RECRD05WAORG,
91100           CONST01
91110           RECRD15ORGDESCRIP,
91120           CONST03
91130           RECRD01WASHIFT,
91140           CONST02
91150           RECRD06WANEXTVACD,
91160           CONST05
91170           RECRD02WANODEPEND,
91180           CONST04
91190           RECRD06WARATE,(X.XXX (,
91200           CONST01
91210           RECRD08WAYTDHRS,(XXXXX.X (,
91220           CONST01
91230           RECRD09WAYTDMONEY,(XXXXX.XX (,
91240           CONST02
91250           RECRD07WAYTDWTAX,(XXXX.XX (,
92010           DTAIL
92020           CARRC
92030           CONST01
92040           RECRD07WAYTDFICA,(XXX.XX (,
92050           CONST02
```

```
92060          RECRD07WASLHRSACC,(XXXX.X (,
92070          CONST02
92080          RECRD06WAVACHRSUD,(XXX.X (,
92090          CONST03
92100          RECRD07WAINSAMT,(XXX.XX (,
92110          CONST02
92120          RECRD07WALOANBAL,(XXX.XX (,
92130          CONST02
92140          RECRD07WAMDEDAMT,(XXX.XX (,
92150          CONST04
92160          RECRD06PCHOURSWK,(XXX.X (,
92170          CONST02
92180          RECRD06PCHOURSPD,(XXX.X (,
92190          CONST02
92200          RECRD09PCGROSS,(XXXXX.XX (,
92210          CONST01
92220          RECRD09PCNET,(XXXXX.XX (,
92230          CONST01
92240          RECRD08PCWTAX,(XXXX.XX (,
92250          CONST01
92260          RECRD07PCFICA,(XXX.XX (,
93010          BREAK01P,
93020               01WAMANNO1,
93030          CARRC  0
93040          CONST15
93050          TOTAL14PCGROSS,(±X,XXX,XXX.XX (,
93060          CONST05
93070          TOTAL12PCWTAX,(±XXX,XXX.XX (,
93080          CONST05
93090          TOTAL12PCFICA,(±XXX,XXX.XX (,
93100          CONST05
93110          TOTAL14WAYTDMONEY,(±X,XXX,XXX.XX (,
93120          CONST05
93130          TOTAL12WAYTDWTAX,(±XXX,XXX.XX (,
93140          CONST05
93150          TOTAL12WAYTDFICA,(±XXX,XXX.XX (,
93160          CONST01
93190CLRRTENTRYCLRPT   IDENT
93200          MODE    AUTOCODER
990100A    9900TABLE   PROGHLT
99020          TITLE   HANDLE OBJECT PROGRAM HALTS
99030          TYPE    (OBJECT PROGRAM HALT ENCOUNTERED (□
99040          HLT     )6000
99045          TR      *-5
990500A    9901TABLE   PROGERR
99060          TITLE   HANDLE 900 SERIES AND ANY AND TAR
99070          TYPE    (900 SERIES,ANY,OR TAR CONDITION (□
99080          HLT     )6001
99090          TR      *-5
99999FINAL     END
```

CHAPTER 4

7080 Decision Table System Specifications

While this chapter is essentially a reference section, the general characteristics of the system are explained initially. Aspects of the system are defined and related to other aspects of the system so that the system can be seen as an entity. This general description is followed by detailed specifications, rules, and restrictions beginning with the caption CONDITIONS.

System Characteristics

The specific operations needed to perform a data processing job are specified through a set of decision tables. These tables reflect the decisions of the system and the actions needed to produce the proper results.

Three distinct procedure languages are permitted in the 7080 Decision Table System.

1. Decision tables
2. Autocoder III pseudo tables
3. Report Writer pseudo tables.

Procedure and operand descriptions are separate. Four operand (data) description tables are used to define the operands used by the program. A single form, the Decision Table Name Description sheet, is used for the description of all operands except the 7080 IOCS File

Table form which is used to define input-output files. Various types of operands fall into each operand description table.

0000 -- I/O FILES -- All input/output files used in the program must be described in this table. This will include the 7080 IOCS File Table Macro-instructions followed by the definitions of the records contained in the file. Field definitions will define the format of the records as they will exist in core storage. (Tape assignment entries will precede the description of the various input/output files.)

0001 -- CORE FILES -- Working storage records are described in this section. Table 0001 is omitted if not needed.

0002 -- MISCELLANEOUS, VARIABLE -- This section is used to define temporary locations for variables, switches, address constants and index pointers.

0003 -- MISCELLANEOUS, FIXED -- This section is used to define arithmetic expressions, symbolic expressions, constants, value lists, and reference lists.

The table form is the primary procedure form accepted by the system. Two basic types of tables are recognized in the 7080 Decision Table System -- "open" and "closed" tables. An open table corresponds to normal programming or open subroutines which may be transferred.

to by GO TO commands and in turn have GO TO's to indicate the next table to be considered. They may also call on closed tables with the DO command. Closed tables correspond to closed subroutines and can only be executed when called by a DO command (corresponds to Autocoder LINK), which controls the entrance <u>and</u> the exit of the routine. A closed table should not have GO TO commands within it (it can have other DO's, however).

Autocoder and Report Writer pseudo tables are auxiliary modes accepted by the system and treated as closed tables. They are written in their normal form with a suitable table header preceding each. Thus, entry is permitted by the "DO table name" in a normal decision table.

In a decision table, conditions are written in the top portion, actions in the lower portion. The condition and action stubs are to the left, the entries to the right. The four quadrants are separated by vertical and horizontal double lines. Table headers pertinent to the entire table are written above the main body of the table.

Conditions, in the upper portion of the table, are written in limited or extended entry form. Based upon the satisfaction of one or more conditions, certain actions specified in the action portion of the table are carried out.

Two categories of conditional statements are recognized by the system: relational, where two values are compared for identity or collating relationship, and state, where values or switches are tested to see if they are in a certain state; e.g., ON, POSITIVE, NUMERIC, ZERO, etc.

Actions, in the lower portion of the table are used to assign values, obtain input data, provide output data, and control the sequence of the program. Actions, like conditions, can be represented in either limited or extended entry form. In limited entry form an X in the entry indicates "execute" while a blank means "do not execute"; while in extended form the second operand and sometimes part of the operator appear in the entry.

A rule consists of one or more conditions followed sequentially by one or more actions that are executed if all the conditions are satisfied. These actions will be executed in the order written.

An analyst may explicitly indicate all possibilities to be covered with a set of rules. When unexpectedly no rules are satisfied, an error table named on the Decision Table Header will be considered.

When all possibilities have not been considered by the analyst, and it is possible that the conditions of none of the rules will be satisfied, an ELSE or "all others" rule number is specified in the table header. This rule number references a rule within this table in which no conditions appear (an unconditional rule), but appropriate actions are stated. An <u>unconditional</u> rule is used when certain cases have been written, but when they do not apply, the unconditional rule is executed. An unconditional rule must, of course, be the last one considered in executing a table.

The general form of the relational statement is:

Operand 1 Operator Operator 2

The operator may be:

EQ stands for	is equal to
UN	is not equal to
GR	is greater than
LR	is lesser than
GE	is greater than or equal to
LE	is lesser than or equal to
VS	versus

Operand 2 can be used in the stub with an appropriate Y, N or blank in each entry (limited entry form) as shown below:

STUB	ENTRY		
	Rule 1	Rule 2	Rule 3
MASTERID EQ DETAILID	Y		N

Or operand 2 may occur in the entry itself (extended form).

STUB	ENTRY	
	Rule 1	Rule 2
MASTERID EQ	DETAILID	NORECORD

VS is used in the limited entry form to compare two values specified in the stub - then in the entry the other appropriate symbols are used for the particular relationship with the implication that the symbol is substituted for the VS in the particular rule; e. g. ;

STUB	ENTRY			
	Rule 1	Rule 2	Rule 3	Rule 4
MASTERID VS DETAILID	GR	GR	LR	EQ

In extended entry the relational operators (EQ, GR. . .) can be in the stub (when the comparison is the same for all rules) or in the entry as shown above.

The general form of the state condition is:

Operand 1   Operator   State

Only two operators exist in this type of condition statement:

IS

NT (is <u>NoT</u>)

Four state categories are recognized:

1. POSITIVE (POS)
   NEGATIVE (NEG)
   (Either the full word POSITIVE/NEGATIVE or the abbreviations POS/NEG may be used.)

2. ON
   OFF

3. ZERO
   BLANK

4. NUMERIC

State conditions, like relational, can be written in either limited or extended entry form.

For relational statements there are rules for comparison, and for both relational and state conditions there are restrictions on the operands or names which can be used. For a more complete explanation see <u>CONDITIONS.</u>

Action operators are concerned with (1) the movement of the value of one field or record to another field or record, (2) the movement of data to and from input/output units, and (3) the control of the sequence in which operations are performed.

Examples of the general forms of actions operators are shown below:

MOVE	Operand 1	TO	Operand 2
SET	Operand 1	EQ	Operand 2
GET	File name		
GO TO	Table name		
DO	Table name		

Actions are spelled out more fully under Input/Output Operators, Assignment Operators, and Sequence Control Operators.

Names

Operations are performed upon specified data which may be a file, a record within a file, an associated group of fields, various types of variable data fields, constants, as well as expressions which cause data manipulations. Each of these is named and defined on one of the four Decision Table Name Description Sheets (0000 . . . 0003).

A name may contain up to 10 characters in any pattern of letters (A - Z) and numbers (0 - 9) with the restriction that a name must start with a letter, and may not contain imbedded blanks, or special characters. Names chosen by the analyst should be descriptive of the value they represent to improve the communicability of the program. Names are always written left justified.

Table Names -- Decision tables are named (or numbered) in the table header which precedes each table. These names are used when the table is referenced in the program.

A table name is constructed like any other name in the system (10 alphameric characters), for example, GROSSTONET. If desired, a table may be given a 4-digit numeric identification in the table header. These tables may be referenced in other tables or within the table itself by affixing this number to the letters TAB, for example, TAB0024, TAB9621, etc. If a table has a name and a number they are considered as synonyms.

A data description table (0000...0003) may be given a table name, but this name may not be referenced from another part of the system.

Named Operands

FILE -- A file is a collection of one or more records associated with an input or output device. The file itself is named and defined on the 7080 IOCS File Table. The records which compose the file are described on data description table 0000.

RNAME -- Record names are assigned in data description table 0000 to identify the logical record (the amount of data delivered by I/O operators), and on table 0001 to identify working storage areas.

GNAME -- Group names permit a continuous section of storage to be given a name. GNAMEs are used in tables 0000, 0001 and 0002.

Variable Field Names -- These are names for signed and unsigned numeric fields, alphameric fields, report fields to be edited, characters used for bit switches, and preassigned value fields. When these fields describe records contained in the I/O files, they are defined in data description table 0000. For working storage records, they are defined in table 0001, and when they define temporary storage locations they are described in table 0002. The classes of variable field names follow:

FLD - The name, length and special characteristics (such as sign or decimal position) of numeric and alphameric fields is specified with this class entry.

PRE - The name, length, special characteristics, and specific values of preassigned variables are specified with this class entry.

RPT - Report field formats are specified with this class entry. Such editing functions as placing commas, decimal points and dollar signs for printed reports are accomplished.

BITCD - One character variables are set up and their 1, 2, 4 and/or A bits named. Each named bit can then be used by the program as an on-off switch.

PCON -- Permanent constants that are never or very seldom changed, such as 3. 1416, are specified in data description table 0003 and classified as PCON.

ADCON -- Address constants are utilized by closed Autocoder pseudo tables and in developing initial base addresses when using index pointers for operation on variable length input or output files (see below).

POINT -- Index pointers are designed in the system as point names for fields containing a 7080 address. The address is controlled by the programmer in manipulating variable length input/output records.

Indirect Address Registers -- Ten pre-defined address constants, tagged X0 through X9, are provided by the system. These may be used to effect address modification in referencing data fields.

ALTSW -- Alteration switches are named with his entry class.

Expressions -- There are two types of expressions: arithmetic and symbolic. These are named and specified in data description table 0003.

MEXP - Arithmetic expressions are specified in much the same manner as for the MATH macro in the Autocoder system. The expression name is referenced in the program, thus calling for the proper computation. Arithmetic expressions are further explained in the description of Table 0003.

SEXP -- Symbolic expressions allow several fields of data to be treated as a single operand. Variable names, constant names, symbolic expression names and literals may be operands in a single symbolic expression. Symbolic expressions are further explained under Table 0003 in this manual.

VLIST -- A value list consists of a list of constant values that can be used in the condition section of a program table. A value list is named in data description table 0003 and described. It can then be named as an operand in a decision table to see if the current value of an input item is contained on the list. Value lists are further explained under Table 0003.

RLIST -- A reference list provides a set of constant arguments and values to be specified by the user. Reference lists are named and described in data description table 0003. It is then possible to determine if a particular argument is in the table and use the function as the operand in an action. Reference lists are described more fully under Table 0003.

Other Operands

The following operands can be utilized in the program without being described on a Decision Table Name Description form:

Literals

A literal is the value itself rather than the name of a value (e. g., the value is expressed in the program, whereas constants are described). Literal values up to 8 characters in length may be used directly in the table -- when more positions are needed they are described in the operand description as a constant. Two types of literals are available -- numeric and alphameric.

Numeric literals must consist of all digits, up to 8 are permitted. A sign may precede the value. An unsigned literal is considered to be positive. If the value is other than an integer (whole number) a decimal point must be included to indicate the number of integers and decimals. Further, there must always be a sign associated with a value containing a decimal point. The sign and point are not part of the literal value itself. No special symbols are needed to indicate that it is a numeric literal.

No signs or decimal points are used with an alphameric literal (the symbols . + and - may be used but they do not connote sign or decimal). It may have up to 8 positions also, but to distinguish it from names of values special symbols must surround it. The single quote (') symbol is used. This symbol is keypunched as #. Examples are:

'LGA'
'1 2 3 - T P S. '

All symbols of the character set may be used except the quote symbol which delimits the literal. A constant which contains this symbol may be described in the operand description along with values that are more than 8 positions in length.

TABSTOP

A program can be halted by using the operand TABSTOP in a GO TO statement, the "next table" position, or the "error table" in the header. This will cause a dead-end halt in the object program. If it is desired to do other than stop at intermediate points or at job completion, other programming must accomplish the waiting loop or whatever is needed.

Special Names

There are a number of "system words" which may be used by the programmer without identifying them in the Decision Table Name Description sheet. The use and meaning of these words is described where they are used in the condition and action portions of decision tables. The words are:

ZERO
BLANK
NUMERIC
NEGATIVE
NEG
POSITIVE
POS
ON
OFF

CONDITIONS

Relational

As enumerated under systems characteristics the relational operators may be:

EQ	stands for	is equal to
UN		is not equal to
GR		is greater than
LR		is lesser than
GE		is greater than or equal to
LE		is lesser than or equal to
VS		versus

Both operands should not be literals, constant names, value list names or reference list names. Operand 1 or operand 2 can be the name of one of the following:

- Variable name
- Variable name, Xn
- Group name
- Group name, Xn
- Arithmetic expression name
- Symbolic expression name
- Constant name
- Literal (the value itself)
- Value list name
- Reference list name

Conditions may be in limited or extended entry form. VS is used in limited entry to compare two values in the stub, with the appropriate operators (above) in the entry. Xn is the designation of one of ten special registers available for address modification (this topic is covered more thoroughly later).

Rules for Comparisons

For numeric values, if the number of decimal and integer positions for both values agree, a simple comparison of values is made. Otherwise, the value with the lower number of decimal positions is zero-filled in its low-order positions until the number of decimal positions in both values agree. The value with the lower number of integer positions is zero-filled in its high-order positions until the number of integer positions for both values agree. For example:

Operand 1	87654.32
Operand 2	321.98765

Three low-order zeros are added to Operand 1 and two high-order zeros are added to Operand 2, giving:

Operand 1	87654.32000
Operand 2	00321.98765

The values are now compared, position by position. Note that all negative numbers are considered smaller than positive numbers. Further, when two negative numbers are compared, the number with the smaller absolute value is considered the larger number.

For alphameric values, if the number of positions in both values agree, a simple comparison is made. Otherwise, the value with the smaller number of positions is filled with blanks in its low-order positions until the number of positions in both values agree. For example,

Operand 1	ABCDEF
Operand 2	ABCD

Two blanks are added to the right of Operand 2, giving:

Operand 1	ABCDEF
Operand 2	ABCDbb

Comparison of the values is then made, position by position, based on the collating sequence.

<u>Conditions: State</u>

Only two operators exist in this type of condition statement:

IS

NT (is NoT)

Only certain operand types can be tested to see if they are in a certain state. Four state categories are recognized:

1. $$\left\{\begin{array}{l}\text{Variable name}\\ \text{Group name}\\ \text{Arithmetic expression name}\end{array}\right\} \left\{\begin{array}{l}\text{IS}\\ \text{NT}\end{array}\right\} \left\{\begin{array}{l}\text{POSITIVE (POS)}\\ \text{NEGATIVE (NEG)}\end{array}\right\}$$

Either the full word POSITIVE/NEGATIVE or the abbreviations POS/NEG may be used. The value of the group name must have only a single conventional sign indicator.

2. $$\left\{\begin{array}{l}\text{Bit switch name}\\ \text{Alteration switch name}\end{array}\right\} \left\{\begin{array}{l}\text{IS}\\ \text{NT}\end{array}\right\} \left\{\begin{array}{l}\text{ON}\\ \text{OFF}\end{array}\right\}$$

3. $\left\{\begin{array}{l}\text{Variable name}\\\text{Arithmetic expression name}\\\text{Symbolic expression name}\\\text{Group name}\end{array}\right\} \quad \left\{\begin{array}{l}\text{IS}\\\text{NT}\end{array}\right\} \quad \left\{\begin{array}{l}\text{ZERO}\\\text{BLANK}\end{array}\right\}$

4. $\left\{\begin{array}{l}\text{Variable name}\\\text{Group name}\\\text{Symbolic expression name}\end{array}\right\} \quad \left\{\begin{array}{l}\text{IS}\\\text{NT}\end{array}\right\} \quad \text{NUMERIC}$

State conditions, like relational, can be written in either limited or extended entry form. State conditions cannot use the address modification feature of the system (X0 - X9).

ACTIONS

Input/Output Operators

The functions of the 7080 IOCS package will be available to the 7080 DTS user. The instructions which call for these functions will be written in the decision tables themselves in the format of 7080 DTS.

The user may use the particular IOCS package that he desires. The definition of the files and the functions required will be included in the operand description portion. In general, the format of the Autocoder system will be used in this regard.

Note: This specification will not attempt to make a precise definition of all the functions of the IOCS package or how they are to be described in the 7080 DTS, or in fact, how it will operate in the object program. (See Preliminary Manual--Input/Output Control Systems for IBM 7080--No. J28-6188). It will suffice to indicate the input/output operators and format that are useable in the decision table itself. Five action operators are recognized in the input/output section.

OPEN... The OPEN command is used to tell IOCS to start the reading of all input files, do the necessary label checking if called for, and get the first block of information into the machine. In the case of output files, label records will be created if desired, and any other functions called for in the IOCS package. No operands are required, since all files will be opened by generated linkage to IOCS.

GET... This action operator is a means of providing the next logical record of the file. Two forms of the command exist:

GET filename

GET filename IN recordname

The first command provides the next record in a working area provided by IOCS. The second command, with two operands, first moves the next logical record into the working area and then a move command is generated to place this information in the area identified by the record name. The connector IN must appear in the action stub, otherwise the second operand (record name) will be ignored. End of file is reached when the last record has already been processed; transfer is made to the open table identified in the file section of the operand description.

Deblocking of records is handled automatically with the GET action. The I/O processor will insure that the next block is read into the machine when a block is depleted. The user need not concern himself about actual read commands.

PUT... The PUT action is the counterpart of the GET action on the output side. Two forms of the command are available.

PUT filename

PUT filename FR (FROM) recordname

With the PUT filename command, the record in the output area is released by the program and goes into the output area to be written out automatically by IOCS when a block has been filled. The user need not concern himself with blocking of output records. The blocking factor itself is specified in the operand description area.

The PUT filename FR recordname first causes the movement of the information of a record area to the standard working storage position of the file, and then to the output blocking area to be written when the block is full. If no blocking is called for, the PUT command will write the record as soon as an I/O device becomes available. The connector FR must appear in the action stub, otherwise operand 2 will be ignored.

CLOSE... This action is used when the program is finished with a particular file. CLOSE must have as operand 1 the name of the file to be closed. This file name must be the name of a GET/PUT type file. In the case of input, after doing any record counting or such called for in the operand description, the tape is merely rewound. On the output side, the last block will be written on the output device, a tape mark and trailing label will be written, and then the tape will be rewound.

TYPE... This command will be available for putting out small amounts of information either on the console typewriter or on the on-line printer. Acceptable operands for this command will be record name, group name, variable name, constant name, literals, and symbolic expression name. An arithmetic expression is not an acceptable operand for TYPE, since there is no way to move a group mark immediately behind the expression to halt the type-out.

Assignment Operators

MOVE... TO... This operator is used in conjunction with two operands to move the value of one field or record to another field or record. Several types of operands can be used, but <u>only</u> the following combinations of operand 1 and operand 2 are allowed:

1. MOVE Record name TO Record name

$$2.\quad \text{MOVE} \left\{\begin{array}{l}\text{Variable name}\\ \text{Group name}\\ \text{Constant name}\\ \text{Literal}\\ \text{Arith. expr. name}\\ \text{Symbolic expr. name}\\ \text{Reference list junction}\end{array}\right\} \text{TO} \left\{\begin{array}{l}\text{Variable name}\\ \\ \text{Group name}\end{array}\right\}$$

$$3.\quad \text{MOVE} \left\{\begin{array}{l}\text{ON}\\ \\ \text{OFF}\end{array}\right\} \quad \text{TO} \quad \text{Bit switch name}$$

$$4.\quad \text{MOVE} \left\{\begin{array}{l}\text{ZERO}\\ \\ \text{BLANK}\end{array}\right\} \quad \text{TO} \left\{\begin{array}{l}\text{Group name}\\ \text{Variable name}\\ \text{Record name}\end{array}\right\}$$

$$5.\quad \text{MOVEV} \left\{\begin{array}{l}\text{Index pointer name, xxxx}\\ \text{Index pointer name}\\ \text{Index pointer name, RM}\\ \text{Index pointer name, Var. name}\end{array}\right\} \text{TO} \quad \text{Record name}$$

$$6.\quad \text{MOVEV} \left\{\begin{array}{l}\text{Record name}\\ \text{Variable name}\\ \text{Constant name}\\ \text{Literal}\\ \text{Expression name}\end{array}\right\} \text{TO} \left\{\begin{array}{l}\text{Index pointer name}\\ \text{Index pointer name, xxxx}\\ \text{Index pointer name, RM}\end{array}\right\}$$

$$7.\quad \text{MOVE} \begin{array}{l} \left\{\begin{array}{l}\text{Variable name}\\ \text{Group name}\end{array}\right\} \text{, Xn} \quad \text{TO} \left\{\begin{array}{l}\text{Variable name}\\ \text{Group name}\end{array}\right\} \\ \\ \left\{\begin{array}{l}\text{Variable name}\\ \text{Group name}\end{array}\right\} \quad\quad\;\; \text{TO} \left\{\begin{array}{l}\text{Variable name}\\ \text{Group name}\end{array}\right\} \text{, Xn}\end{array}$$

Operand 1 is always the name of the sending area and operand 2 the name of the receiving area.

1. Record Movement

If records are to be moved, both operands must have been defined with the RNAME class code. All I/O records are a multiple of 5 characters (an IOCS requirement) and so must be internal records defined for working storage if they are to be moved with the MOVE record name action. Also, they must end with a record mark since high-speed transmit will be used.

Note: If a record does not have these characteristics, the record can, in addition, be defined as a group (GNAME) and then a MOVE group name TO group name can be utilized.

2. Field Movement

The second type of MOVE, the most common, is used to move values from one field to another. The general form and the permissible operands are shown below:

$$\text{MOVE} \left\{ \begin{array}{l} \text{Group name} \\ \text{Variable name} \\ \text{Constant name} \\ \text{Literal} \\ \text{Arithmetic expr. name} \\ \text{Symbolic expr. name} \\ \text{Reference list function} \end{array} \right\} \text{TO} \left\{ \begin{array}{l} \text{Variable name} \\ \\ \text{Group name} \end{array} \right\}$$

The MOVE group name TO group name command operates in much the same way as the MOVE record name in that the characters are peeled off from the left of the sending field and placed in the receiving field starting from the left.

The transmission of characters is stopped when the end of the shorter area has been reached. This is comparable to the normal Autocoder move. When the longer area is the sending area, the excess characters are truncated. When the shorter area is used as the sending area, the excess characters of the resulting field are unaffected.

When information is being moved into a variable field, special rules will be in effect. Again, the shorter of the two operands, as specified in the operand description tables, delimits movement of data. However, when <u>numeric</u> data is being moved and operand 1 is the shorter field, low-order decimal positions and high-order integer positions in the receiving field are zero-filled up to the number of decimal and integer positions specified in the receiving field. For example:

Before the MOVE... TO... action:

Operand 1	987.65
Operand 2	6 integer and 4 decimal positions specified

After the MOVE... TO... action:

Operand 1	987.65
Operand 2	000987.6500

When operand 2 is the shorter field, low-order decimal positions and high-order integer positions of operand 1 are truncated to satisfy the number of decimal and integer positions specified for operand 2. Before low-order truncation, half adjustment will take place. For example:

Before the MOVE... TO... action:

Operand 1	98765.4321
Operand 2	4 integer and 2 decimal positions specified

After the MOVE... TO... action:

Operand 1	98765.4321
Operand 2	8765.43

When alphameric data is being moved and operand 2 is the longer field, low-order positions of operand 2 are blank filled. For example:

Before the MOVE... TO... action:

Operand 1	ABCD
Operand 2	6 positions specified

After the MOVE... TO... action:

Operand 1	ABCD
Operand 2	ABCDbb

When operand 2 is the shorter field, low-order positions of operand 1 are truncated. For example:

Before the MOVE... TO... action:

Operand 1	ABCDEF
Operand 2	4 positions specified

After the MOVE... TO... action:

Operand 1	ABCDEF
Operand 2	ABCD

Numeric data can be moved either to a numeric field or to an alphameric field. When moved to an alphameric field, the numeric field is treated as though it were an alphameric field. Alphameric data can be moved only to an alphameric field. When numeric data is moved, the sign of the sending field (operand 1) is moved to the receiving field (operand 2). When operand 1 is an expression name, the value of the expression is computed, and then this value is moved to the receiving field.

When operand 1 is a numeric literal, its decimal and integer length is established as written. The rules for zero filling and truncation of the receiving field are the same as for other types of operands in the sending field. Note that the total number of decimal and integer positions for a numeric literal cannot exceed 8. When operand 1 is an alphameric literal, its length is established as written. The rules for blank filling and truncation of the receiving field are the same as for other types of operands in the sending field. Note that the number of positions for an alphameric literal cannot exceed 8.

When the JOIN operator of a symbolic expression is used to develop a field to be moved to another field, it will be treated as a single value and hence, the rules for blank fill will apply. Even if the value developed is all numeric, it will be treated as an alphameric value.

A special situation exists in the case of moving a function value from a reference list. In the condition area the argument and the table name are identified and a search is made to see if that argument exists in the table. If it does, the corresponding value is moved to the location indicated in the first line of the table definition. This name can then be used in the action portion of the table to obtain the value. If the user has not tested to find out if the argument is in the table in the condition area and he calls for the movement of that data in the actions, the last value placed in this position will be moved. It is the responsibility of the user to insure that the proper value is there.

3. Setting of Bit Switches

The third kind of MOVE sets a Bit switch ON or OFF.

$$\text{MOVE} \quad \left\{ \begin{matrix} \text{ON} \\ \text{OFF} \end{matrix} \right\} \quad \text{TO} \quad \text{Bit switch name}$$

The MOVE ON/OFF command can use only a Bit switch name as operand 2. The effect of this is to turn a Bit on or off such that it can later be tested in the condition area or can be part of the output record.

4. Zero or Blank Fill of Operands

The fourth type of MOVE is used to zero or blank-fill an operand named in the second operand position.

$$\text{MOVE} \quad \left\{ \begin{matrix} \text{ZERO} \\ \text{BLANK} \end{matrix} \right\} \quad \text{TO} \quad \left\{ \begin{matrix} \text{Variable name} \\ \text{Group name} \\ \text{Record name} \end{matrix} \right\}$$

The system will automatically generate the number of zeros or blanks required to fill the receiving field. Anything currently in the receiving field will be erased.

5. Variable Length Data Movement

The fifth type of MOVE is used to move information out of a variable length input record. It will be the responsibility of the programmer to work through an input record obtaining the pertinent input values and manipulating them as required. Index pointers can be set up in the system to assist the user in this regard. The pointer, in reality, is a 6-position signed field containing an expanded 7080 address. This address may be initialized (presumably to the beginning of the variable length portion of the record).

$$\text{SET} \qquad \text{Index pointer name} \quad \text{EQ} \begin{Bmatrix} \text{Field name} \\ \text{Adcon name} \end{Bmatrix}$$

The address is then manipulated as trailer items and fields are moved out to a working storage area where they can be addressed directly when the particular item is identified. The value of the index pointer is maintained by the programmer using the appropriate arithmetic. The pointer may be advanced by the length of the item just moved to be ready for the next item, or if the record itself contains size or number of occurrences information, these values can be used to modify the index.

The MOVE operator is suffixed with a V (variable) to indicate that this is a special move operation.

$$\text{MOVEV} \begin{Bmatrix} \text{Index pointer name} \\ \text{Index pointer name, xxxx} \\ \text{Index pointer name, RM} \\ \text{Index pointer name, Variable name} \end{Bmatrix} \text{TO} \quad \text{Record name}$$

The xxxx can control the number of characters, or the record terminating record mark can control the movement (RM option).

6. Fabrication of a Variable Length Output Record

The sixth type of MOVE is used to fabricate a variable length output record. This is also described under Table 0000 in the operand description portion of this document.

```
       (Record name    )      (Index pointer name                  )
       |Variable name  |      |Index pointer name, xxxx            |
MOVEV {Constant name    } TO {                                      }
       |Literal        |      |Index pointer name, RM              |
       (Expression name)      (Index pointer name, Var. name       )
```

When information is to be moved, and only an index pointer name is given as operand 2, the length of the sending area determines size. When the xxxx option is used, xxxx indicates the number of characters. The RM option will transmit data until a record mark in the sending area stops transmission (sending area must be multiple of 5 and end in a 4 or 9 position). If a variable name (signed or unsigned) is used, the number found in the variable name field will control the number of characters moved. It is, of course, the responsibility of the programmer to maintain the proper value of the index pointer.

7. Generalized indexing

The seventh type of MOVE permits the use of predefined address registers (X0 through X9) for effecting address modification. The 7080 Decision Table System utilizes the indirect addressing facility of the 7080 to accomplish this. Indirect addressing allows address modifications without changing the addresses of instructions within the decision table, and allows the DTS to remain non-introspective; i.e., the DTS does not modify its own program.

$$\text{MOVE} \begin{Bmatrix} \text{Field name, Xn} \\ \text{Group name, Xn} \end{Bmatrix} \quad \text{To} \quad \begin{Bmatrix} \text{Field name, Xn} \\ \text{Group name, Xn} \end{Bmatrix}$$

In operation the 7080 DTS will supply the programmer with 10 tagged pre-defined address constants (ADCON's) which are used to hold the addresses to be modified. They are tagged X0 through X9, and are referred to as Xn in the description that follows:

Initialization: Any of the Xn may be set to an initial address by the SET or MOVE action command.

$$\text{SET} \quad \text{Xn} \qquad\qquad \text{EQ} \begin{Bmatrix} \text{Field name} \\ \text{Adcon name} \end{Bmatrix}$$

$$\text{MOVE} \begin{Bmatrix} \text{Field name} \\ \text{Adcon name} \end{Bmatrix} \qquad \text{TO} \quad \text{Xn}$$

In the case of Field name the initial address will reference the right hand end of the data field; when Adcon name is used, the ADCON must contain a right hand address.

Modification of Registers: Any Xn, once initialized, may be incremented or decremented by the SET or MOVE command (see two-address arithmetic).

$$\text{SET}\quad \text{Xn}\qquad \begin{matrix}+\\-\end{matrix}\ \left\{\begin{matrix}\text{Field name}\\ \text{Arith. expression}\\ \text{Literal}\end{matrix}\right\}$$

$$\text{MOVE}\ \left\{\begin{matrix}\text{Field name}\\ \text{Arith. expression}\\ \text{Literal}\end{matrix}\right\}\ \begin{matrix}+\\-\end{matrix}\qquad \text{Xn}$$

Note that the *, / (multiply, divide) operators may not be used in stepping Xn.

The amount of the increment or decrement is determined by the number of characters from the current right hand address to the desired right hand address.

Using the Address Registers: Any Xn may be used to supply the address portion of a command in either the condition or action section of a procedure table, as follows. Both Operand 1 and Operand 2 may be modified.

MOVE	FLD 1, X2	TO	FLD 2, X6
SET	FLD 3, X2	EQ	ZERO

The Address Registers can also be used in the condition portion of a table as noted earlier.

FIELD 1, X1	EQ	FIELD 2, X2
TAG 2, X4	GR	ZERO

In each case, the field name preceding Xn supplies the <u>format</u> of this operand, and the Xn supplies the <u>address</u> of this operand. (It should be noted that the format field will usually be the same data name used to initialize Xn.) Therefore, only the address part of a command can be modified. The format (size) of an operand is fixed.

In any cases in which the MOVE or SET verb results in instructions which require other than right hand addresses, 7080 DTS will automatically adjust Xn during execution. Xn's value will <u>not</u> be changed itself, only its value with regard to the instruction requiring the adjustment. Therefore, the programmer need only be concerned with right hand locations throughout a tabular program.

Restrictions: The address modification feature may <u>not</u> be used with any of the input-output or sequence control actions or with the following operand types:

<u>OPERATOR</u>	<u>OPERAND</u>
MOVE, SET	Any <u>receiving</u> field defined as a <u>RPT</u> field.
MOVEV	Neither. MOVEV is a special purpose action developed exclusively for moving <u>variable length</u> segments. It can be used with a defined <u>POINT</u> only.

MEXP, SEXP — No member of a math or symbolic expression may contain an indirect reference. In addition, since expressions are not generated adjacent to each other (even though defined one after another), address modification may not be used to step from one expression to another.

SET... EQ...

The SET... EQ... command is the direct counterpart of the MOVE command. The MOVE command moves information specified in operand 1 and moves it to the location named in operand 2; the SET moves information named in operand 2 to the location named in operand 1. Otherwise, all rules for the SET... EQ... operator are as defined above with the MOVE command, except that there is no SET V action operator.

1. SET Record name EQ Record name

$$2. \quad \text{SET} \left\{ \begin{array}{l} \text{Variable name} \\ \text{Index pointer name} \\ \text{Group name} \end{array} \right\} \text{EQ} \left\{ \begin{array}{l} \text{Variable name} \\ \text{Group name} \\ \text{Constant name} \\ \text{Literal} \\ \text{Arithmetic expr. name} \\ \text{Symbolic expr. name} \\ \text{Reference list function} \end{array} \right\}$$

$$3. \quad \text{SET Bit switch name EQ} \left\{ \begin{array}{l} \text{ON} \\ \text{OFF} \end{array} \right\}$$

$$4. \quad \text{SET} \left\{ \begin{array}{l} \text{Variable name} \\ \text{Group name} \\ \text{Record name} \end{array} \right\} \text{EQ} \left\{ \begin{array}{l} \text{ZERO} \\ \text{BLANK} \end{array} \right\}$$

Two-Address Arithmetic Operators

Both the MOVE and the SET commands can incorporate two-address arithmetic. The following arithmetic operators are permitted: + (punched &) for addition, - for subtraction, * for multiplication, / for division. The general form of the SET is as follows:

$$\text{SET} \quad \text{Variable name} \begin{Bmatrix} + \\ - \\ * \\ / \end{Bmatrix} \begin{Bmatrix} \text{Variable name} \\ \text{Literal} \\ \text{Constant name} \\ \text{Arithmetic expression name} \end{Bmatrix}$$

SET	A + B	Means	A ← A + B
SET	A - B		A ← A - B
SET	A * B		A ← A * B
SET	A / B		A ← A / B

The operands used for this action can be numeric values only. Algebraic rules are followed when considering the sign of the action operator in conjunction with the signs of the two operands. In all cases, operand 2 operates on operand 1 with the appropriate arithmetic operator and the value is then stored in the field named in operand 1.

The integer and decimal lengths of the receiving field, operand 1, determine the number of integer and decimal positions in the total resulting from the arithmetic. Thus, the decimal positions in operand 2 in excess of the number specified for operand 1 are first half-adjusted and then truncated. High-order integer positions that result are truncated before the value is stored in operand 1. Any carry from the high-order position of operand 1 is truncated; thus, if the user desires to get overflow control he must use the arithmetic expression which gives him an option for overflow protection.

The same rules apply for the MOVE command in doing two-address arithmetic except that operand 2 is the receiving field and operand 1 is the sending field.

$$\text{MOVE} \begin{Bmatrix} \text{Variable name} \\ \text{Literal} \\ \text{Constant name} \\ \text{Arithmetic expression name} \end{Bmatrix} \begin{Bmatrix} + \\ - \\ * \\ / \end{Bmatrix} \text{Variable name}$$

MOVE	A + B	Means	B + A → B
MOVE	A - B		B - A → B
MOVE	A * B		B * A → B
MOVE	A / B		B / A → B

<u>Sequence Control Operators</u>

GO TO... This operator is used to specify the next table to be executed. It can be thought of as an unconditional transfer. The GO TO command can only reference an open table (one never executed under the control of a DO - see below). This action can also reference the special operand TABSTOP, which causes the program to come to a dead-end halt. Three forms are available:

GO TO Table name

GO TO TABnnnn

GO TO TABSTOP

The user may refer to a table either by name or by its number (if both are given in the header they are considered synonymous) with the special prefix TAB for table. Tables are named and/or numbered in the table header. The normal 'next table' or special 'error table' can be specified in the table header; when 'next table' has an entry and the rule also has an entry, the rule GO TO will take precedence.

The GO TO action <u>must</u> be the last action executed in any rule. GO TO must be written in the stub, the table name or number may be written in either the stub or in the entry portion of the table.

DO. . . This operator may only refer to a closed table. The table referenced may be a closed decision table, a closed Report Writer pseudo table or a closed Autocoder pseudo table. Report Writer and Autocoder sections can only be executed by a call from a decision table, utilizing the DO command. (The DO action can be thought of as an RCV (exit point) TSL (entrance point)).

The general form is:

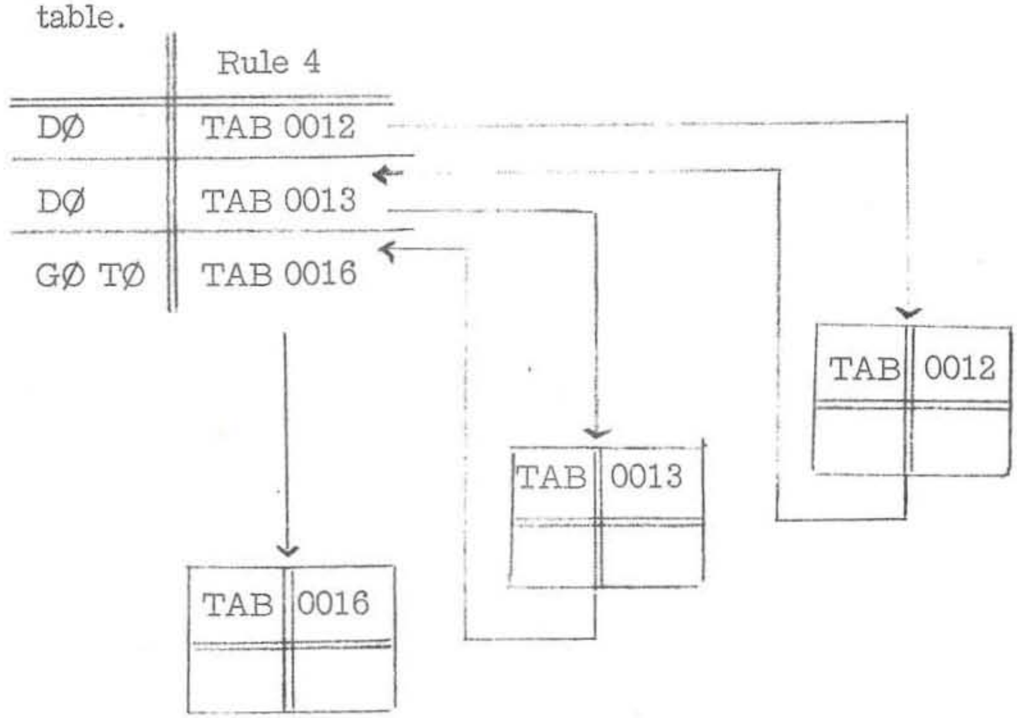
DO Table name

DO TABnnnn

The DO action calls for the execution of another table, entering at the top (decision tables can only be entered at the beginning) and returning after the last action of the rule whose conditions were satisfied has been carried out. Closed tables (those executed under the control of a DO) can also be unconditional (1 rule) tables for carrying out a single procedure. Since the DO command also controls the return to the original table containing the DO, the 'done' table cannot have any GO TO commands in it. A DO table is noted by placing the letters 'DO' in the 'next table' portion of the header.

It is permitted, however, for the 'done' table itself to have DO's of its own calling on other tables or pseudo tables. This is called 'nesting.' It is important that a table <u>not</u> 'DO' itself or any table above it in the nest. Any level of nesting is permitted if the above rules are adhered to <u>and</u> the user can keep it straight in his own mind.

After a table has been executed, control returns to the calling table (the one with the DO command in it) at the action following the DO. If the DO is the last action of a closed table, control will revert to its calling table. If it is the last action of an open table, the 'next table' will be used to direct control to the next table. The illustration below shows the action portion of a rule calling upon two tables and then transferring to another table.



The sequence of events is: execute table 0012, return, execute table 0013, return, transfer to table 0016.

A 'next table' may not be specified in a closed table since this is effectively a GO TO command, which is not permitted. Also, an 'error table' may not be in a closed or 'done' table since control might also be lost. The unconditional (ELSE) rule should always be specified in cases where all possibilities are not covered by the rules.

In summary, the GO TO acts like a normal unconditional transfer to another open decision table (not a Report Writer or Autocoder pseudo table). The DO acts like a RCV - TSL or LINK macro to a closed subroutine. Certain characteristics, as noted above, pertain to the nature of the closed subroutine. DO must always be written in the stub; the table name or number can appear in either the stub (limited form) or in the entry (extended form).

Decision Table Headers

The Decision Table Logic Sheet has four header entries--the first, the table header, is required; the other three are optional. Asterisks indicate that the entry must be filled in.

Table Header

*	Page 1 - 2	Page is written in conventional Autocoder form. Page may be either numeric or alphameric, and written in ascending order.

IBM 7080

DECISION TABLE LOGIC SHEET

PROGRAM IDENTIFICATION	75	80

PAGE (01–02)	LINE (03–05)	CC (06)	TYPE (07)	NO. RULES (08–09)	ELSE (10–11)	TABLE NO. (12–15)	IDENTIFICATION (16–20)	TABLE NAME (21–30)	NEXT TABLE NAME (31–40)	ERROR TABLE NAME (41–80)
	000	0					TABLE			

PAGE (01–02)	LINE (03–05)	CC (06)	(07–11)	(12–15)	IDENTIFICATION (16–20)	ORDER OF CONDITION ROWS (21–52)
	000	1		CONDORDER		

PAGE (01–02)	LINE (03–05)	CC (06)	(07–11)	(12–15)	IDENTIFICATION (16–20)	ORDER OF RULES (21–74)
	000	2		RULE	ORDER	

PAGE (01–02)	LINE (03–05)	CC (06)	(07–15)	IDENTIFICATION (16–20)	COMMENT (21–74)
	000	3		TITLE	

PROGRAM ________ DATE ________

PREPARED BY ________ PAGE ________

NEXT PAGE ________

START NEXT CARD: Dup. Col's. 1–5, Punch B in Col. 6, Skip Col's. 7–33, Dup. Col's. 75–80.

PAGE (01–02)	LINE (03–05)	CC (06)	ACTION OPERATOR (07–11)	NAME 1 (12–21)	OP (22–23)	NAME 2 (24–33)	RULE 1: 01–06 (OP, NAME 2)	RULE 2: 07–12 (OP, NAME 2)	RULE 3: 13–18 (OP, NAME 2)	RULE 4: 19–24 (OP, NAME 2)	RULE 5: 25–30 (OP, NAME 2)	RULE 6: 31–32 (OP, NAME 2)
	01	A										
	02	A										
	03	A										
	04	A										
	05	A										
	06	A										
	07	A										
	08	A										
	09	A										
	10	A										
	11	A										
	12	A										
	13	A										
	14	A										
	15	A										
	16	A										
	17	A										
	18	A										
	19	A										
	20	A										
	21	A										
	22	A										
	23	A										
	24	A										
	25	A										

* Line 3 - 5 Line identification is written in conventional Autocoder form. All header line numbers are 000.

* Card Continuation (CC) 6 Zero in the table header. Columns 1 - 6 are used for sequence checking in 7080 DTS.

* Type Two identification letters can be specified under this column head which designate:

L - limited entry table (when using 2-column entry columns)

M - mixed entry table (when using 12-column entry columns)

* No. Rules 8 - 9 A count of the number of rules, including the "ELSE" rule, is entered here. A maximum of 32 rules is permissible.

ELSE 10 - 11 The rule number of the rule which is to be executed when none of the other rule conditions are satisfied is entered here.

Table No. 12 - 15 If desired, a table may be given a 4-digit numeric identification and specified in this area of the table header. These numbers may be referenced in other tables or within the table itself by affixing this number to the letters TAB, for example, TAB0024. If a table has a name and a number they are considered as synonyms.

Identification 16 - 20 Always TABLE.

Table Name 21 - 30 A 1 to 10 character name. This name is used to reference this table from some other table or from within the table itself. A decision table must have either a table name or a table number.

Next Table Name 31 - 40 For "open" tables this may contain the identification of the next table to be considered. If a rule has an explicit GO TO, the GO TO operand will override the "next table" designation.

For "closed" tables, the letters "DO" <u>must</u> appear in this field.

Error Table Name 41 - 50 The name of the table to be transferred to when none of the rules is satisfied. Will never be used if there is an ELSE rule specified. An "error table" cannot be specified for "closed tables"; the ELSE rule should be used.

51 - 74 Not used.

Program Identification 75 - 80 Normal Autocoder usage.

Order of Condition Rows Header

Page and Line 1 - 5 Same as above.

Card Continuation 6 Always 1.

Order 21 - 52 The order in which the programmer wants the condition rows re-ordered for efficiency reasons. In general, for execution speed, the most likely to fail should be placed at the top; for least space, the rows with the most blanks (not pertinent) should be placed toward the bottom. Starting in column 21, two-digit numbers corresponding to the row number are written indicating the order. For example: If

01	A	EQ	B	Y	N		
02	C	LE	D			Y	N
03	E	EQ	F	Y	Y	N	N

is reordered

03	01	02	
21 22	23 24	25 26	27

the table would be compiled as though it were written,

E	EQ	F	Y	Y	N	N
A	EQ	B	Y	N		
C	LE	D			Y	N

A maximum of 16 condition rows may appear in a single table.

53 - 74 Not used.

75 - 80 Program identification.

Order of Rules Header

Page and Line 1 - 5 Same as above.

Card Continuation 6 Always 2.

12 - 20 RULEORDER

Order of Rules 21 - 74 The rules may be re-ordered by specifying a rule order for efficiency reasons. The two-digit rule numbers are listed in the order in which they are to be considered. For execution speed, the high-frequency rules should be moved to the left. The ELSE rule must always appear last since it is unconditional. If all rules are not re-ordered, the remaining rules will be taken in the order in which they are written. A maximum of 32 can exist in a single decision table.

75 - 80 Program identification.

Comment Header

Page and Line 1 - 5 Same as above.

Card Continuation 6 The first Comment card is 3; if others are needed, the next would be 4, and so on up to 9.

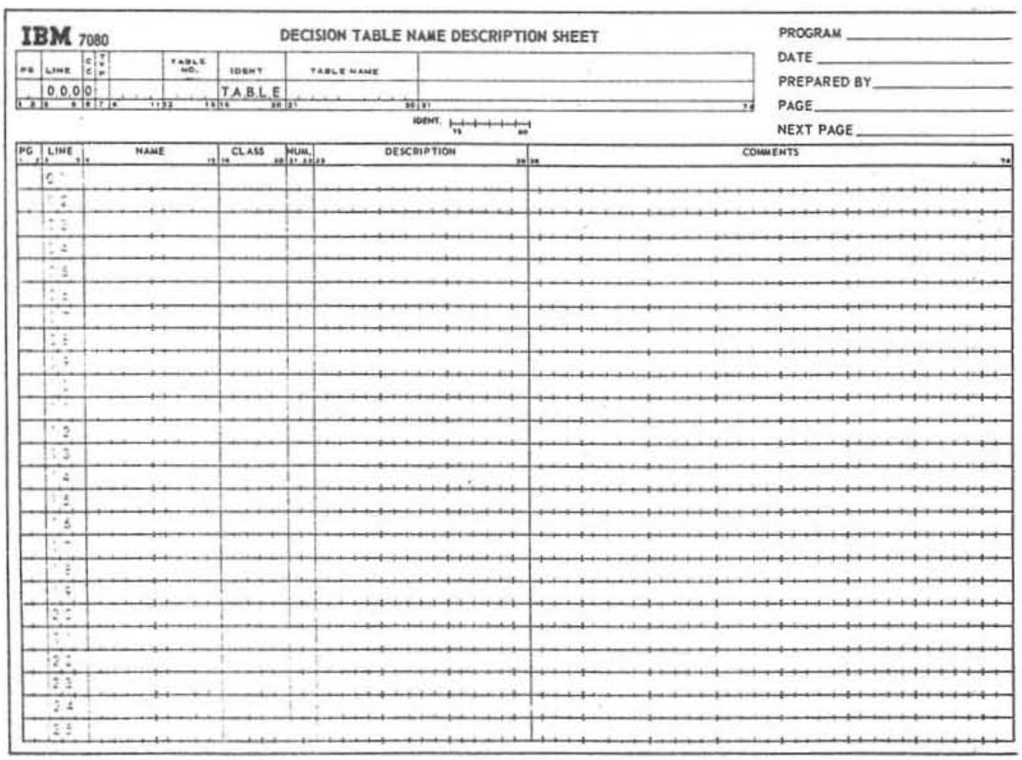
16 - 20 TITLE

Comment 21 - 74 Any remarks or comment. The comment will appear on the decision table listing.

75 - 80 Program identification.

NAME DEFINITION

Names are assigned to operands to be used in the program. These operands may be input/output files, records and fields, working storage items, expressions, etc. The Decision Table Name Description sheet is used to define these operands. This sheet is also used for report specification and Autocoder pseudo tables. A suitable table header will precede and identify each of these sections, the TYPE entry will indicate which kind (D for tables 0000, 0001, 0002, 0003, R for reports, and A for Autocoder). The form is shown below:


IBM 7080 DECISION TABLE NAME DESCRIPTION SHEET

PROGRAM
DATE
PREPARED BY
PAGE
NEXT PAGE

PG	LINE	NAME	CLASS	NUM.	DESCRIPTION	COMMENTS

FORM X22-6913 PRINTED IN U.S.A.

7080 IOCS FILE TABLE MACRO-INSTRUCTIONS (LONG FORM)

PROGRAM: ________ CODED BY: ________ DATE: ________

IOFTA PAYMASTER ◻ 000 ◻ IORETURNTO ◻ IORETURNTO ◻ ~~IORETURNTO~~ MASTERINOF ◻

HI ~~HI~~ ~~LO~~ ~~LO~~ ◻ ~~0~~ 1 ~~2~~ ~~3~~ ~~4~~ ◻ RWD1 ~~RWD2~~ ~~RUN1~~ ~~RUN2~~ ◻ ~~EORb~~ ~~EOFb~~ ~~BOTH~~ NONE ◻ STANDARD ~~NONSTAND~~ ~~NOHEADER~~ ◻ TMbb ~~NOTM~~ ◻ STANDARDb ~~SRECSTANb~~ ~~NONSTANDb~~ ~~NOTRAILER~~ ◻ HSKRWD ~~NORWDb~~ ◻

PRIME ~~NOPRI~~ ◻ SINGLE ~~MULTIb~~ ◻ SEQUEN ~~NONSEQ~~ ◻ ~~CKPTRCDSb~~ NOCKPTRCD ◻ ~~DUMP~~ NODP ◻ ~~DELAYOPEN~~ ~~NODELAYbb~~ ◻

IOFTB IP ~~OP~~ ◻ DATA ~~WORK~~ ~~PROG~~ ◻ ◻ ◻ ~~INITb~~ STACK ◻ CKLNG ~~NOCHK~~ ◻ CMPCK ~~NOCMP~~ ◻ ◻ ◻ GENAREA ~~AREACTV~~ ◻ F ~~V~~ ◻ 150 ◻ 1500 ◻

GETb ~~PUTb~~ ~~BOTH~~ ◻ MASTERIN NOWORKAREA ◻ ~~PAD~~ ~~NOPA~~ ~~00~~ NONE ◻ ~~BOTH~~ ~~DELET~~ EOF9S ~~NONE~~ ◻ ~~BSDb~~ ~~WORK~~ ~~WBSD~~ NONE ◻

~~IOFTC~~ ◻ ~~CKLNG~~ ~~NOCHK~~ ◻ ~~CMPCK~~ ~~NOCMP~~ ◻ ~~OP~~ ~~NO~~ ◻ ~~F~~ ~~V~~ ◻ ◻

Table 0000 - Input/Output Files

A completed table header line (table 0000) must precede the first input-output file description. Subsequent input-output file definitions are not preceded by a header.

Each file is described for IOCS using the long or short version of the IBM IOCS File Table form. A segment of the program in Chapter 3 is used to illustrate the use of the form. Note that lines 03020 through line 03060 correspond to the I/O File Tables A and B on the 7080 IOCS File Table macro-instruction form shown on the opposite page.

```
010200D       0000TABLE   IODEFIN
01030             IOCS
01035             LASN    @500
01040             IOTA    INMASTER□2000□2001□
01050             IOTA    PAYREG□2002□2003□
01060             IOTA    PAYCHECKS□2004□
01070             IOTA    TERMCHECKS□2005□
01080             IOTA    INTRANS□2100□2101□
01090             IOTA    OUTMASTER□2102□2103□
01100             IOTA    EXCEPTIONS□2104□
01110             IOTA    MISCRPTS□2105□
01120             IOTA    END□
03005             LASN    @30000
03010             TITLE   FILE TABLE FOR INPUT MASTER
03020INMASTER     IOFTA   PAYMASTER□000□IORETURNTO□IORETURNTO□MASTRINEOF□
03030                     HI □1□RWD1□NONE□STANDARD□TM  □STANDARD □HSKRWD□
03040                     PRIME□SINGLE□SEQUEN□NOCKPTRCD□NODP□□
03050             IOFTB   IP□DATA□9□STACK□CKLNG□CMPCK□4□GENAREA□F□150□1500□
03060                     GET □MASTERIN□NONE□EOF9S□NONE□
```

Note that table 0000 also contains input-output tape assignments (IOTA) and may also contain I/O tape assignments for special functions (IOTS). (See Preliminary Manual--Input/Output Control Systems for IBM 7080--No. J28-6188.)

Each IOCS file definition is followed by a description of the records which compose the file. Records with different formats may be described with suitable names assigned to fields and groups of fields. The standard Decision Table Name Description Sheet is used for this purpose. A table header line is not required here, as only one is required per section (0000, 0001, ...). A segment of the program in Chapter 3 which is preceded and followed by IOCS file descriptions is used to illustrate IOCS file definitions--lines 05020 through 06040.

Table 0000 must be preceded by an ENT80 macro, to indicate to the 7058 processor that the program will operate in 7080 mode. An Autocoder table will generally precede table 0000 giving appropriate assembly commands such as NOSTP, EEM, ASU, etc., and finally, a table name or number indicating where the execution of the object program is to start, e.g., TR TAB 0005. (See <u>7058 Processor General Purpose Macro Instructions</u> - No. C28-6130.)

```
05010              FILE   TAPE 2000,2001  150 CHAR MASTER RECORD FIXED INPUT
05020MASTERIN     RNAME  AMIEND
05030MIMANNO      FLD    5N               MAN NUMBER
05040MINAME              17A&             MAN NAME
05050MISTATUSCD          1A               STATUS CODE,A-ACTIVE,T-TERMINATED
05051                                     I-INACTIVE

05060MIORG               5N               ORGANIZATION NUMBER
05070MISHIFT             1N               SHIFT EMPLOYEE WORKS
05080MISOCSECNO          9N               SOCIAL SECURITY NUMBER
05090MISEX               1A&              SEX OF EMPLOYEE
05100MIBIRTHDAT          6N               BIRTH DATE
05110MIHIREDATE          6N               HIRE IN DATE
05115MINEXTVACD          6N               NEXT VACATION ELIGIBILITY DATE
05120MIDEDCODE BITCD                      DEDUCTION CODE
05121MIINSDED            1                INSURANCE
05122MILOAN              2                LOAN
05123MIMISC              4                MISC DEDUCTIONS
05130             FLD    1A&              BLANK
05140MINODEPEND          2&               NO OF DEPENDANTS
05150MIRATE              4#&01.03         RATE
05160MIYTDHRS            6#&05.01         YEAR-TO-DATE HOURS
05170MIYTDMONEY          7#&05.02         Y-T-D MONEY
05172MIYTDPDWK           7#&05.02         Y-T-D PAID WORKED TIME
05174MIYTDPDUWK          7#&05.02         Y-T-D PAID UNWORKED TIME
05180MIQTDMONEY          7#&05.02         QUARTER-TO-DATE MONEY
05190MIYTDWTAX           6#&04.02         Y-T-D WITHOLDING TAX
05200MIYTDFICA           5#&03.02         Y-T-D FICA
05210MISLHRSUSD          5#&04.01         SICK LEAVE HOURS USED
05220MISLHRSACC          5#&04.01         SICK LEAVE HOURS ACCUMULATED
05230MIVACHRSUD          4#&03.01         VACATION HOURS USED
05240MIVACHRSAC          4#&03.01         VACATION HOURS ACCUMULATED
05250MIINSAMT            5#&03.02         INSURANCE AMOUT TO DEDUCT
06010MILOANPAY           5#&03.02         PAYMENT AMOUNT FOR LOAN
06020MILOANBAL           6#&04.02         BALANCE OF LOAN
06030MIMDEDAMT           5#&03.02         AMOUNT OF MISC DEDUCTIONS
06040MIEND               1#&03.02         AMOUNT OF MISC DEDUCTIONS
08010              TITLE  FILE TABLE FOR OUTPUT MASTER
08020OUTMASTER     IOFTA  PAYMASTER□030□IORETURNTO□IORETURNTO□IORETURNTO□
08030                     HI □1□RWD1□NONE□STANDARD□TM  □STANDARD □HSKRWD□
08040                     NOPRI□SINGLE□SEQUEN□NOCKPTRCD□NODP□□
08050              IOFTB  OP□DATA□9□STACK□CKLNG□CMPCK□4□GENAREA□F□150□1500□
08060                     PUT □MASTEROUT□PAD9□NONE□NONE□
09010              FILE   TAPE 2102,2103  150 CHAR MASTER FIXED OUTPUT
09020MASTEROUT     RNAME  AMOEND
09030MOMANNO       FLD    5N              MAN NUMBER
09040MONAME               17A&            MAN NAME
09050MOSTATUSCD           1A              STATUS CODE,A-ACTIVE,T-TERMINATED
09060                                     I-INACTIVE
09070MOORG                5N              ORGANIZATION NUMBER
09080MOSHIFT              1N              SHIFT EMPLOYEE WORKS
09090MOSOCSECNO           9N              SOCIAL SECURITY NO.
09100MOSEX                1A&             SEX
```

A description of the entries used in describing input-output files is given below. This information follows that given in the IOCS definition and appears on the Name Description sheet.

FILE

As each file is to be described there must be a class code FILE. The coding form is filled out as follows:

NAME - The file name given in the IOCS File sheet is written in the Name field.

CLASS - The Class code for this entry is FILE.

NUM - The number of different types of records that are going to be described is specified here. If all the records in the file have the identical format, this entry should be left blank or have the entry 01. If more that one record type is to be defined, the appropriate number is specified, e.g., 02, 03 ... The processor will expect this number of RNAME class entries (see below) to follow before the next FILE class appears.

DESCRIPTION - The information specified in the description portion is for documentation purposes only. It is suggested that the I/O media (tape unit number, etc.), maximum record size, etc., be written in this area. See sample above.

The FILE class entry is converted to an Autocoder TITLE card before it goes to the Autocoder assembly system.

RNAME

The record name (RNAME) entry is used to identify the logical record (the amount of data delivered by the GET and PUT commands). This entry allows different formats to be associated with a file area;

there will be a RNAME entry for each different type of record in a file. The name can be referred to by certain input-output and assignment operators, thus allowing the entire record to be operated on.

NAME - A standard name is assigned. If no reference is to be made to the record in the program, the name may be left blank, unless it is the first of two or more RNAMEs being defined.

CLASS - RNAME (record name)

NUM - This entry is used to specify the starting position for this record, i.e., 0 for 0 or 5, 1 through 4 for an address ending in 1 - 4 or 6 - 9. If blank, the next available position will be used. If an area is being redefined for more than one record type, the start position must be the same for all records. The NUM of the first RNAME in a file will be used to locate all subsequent RNAMEs in that file.

DESCRIPTION - The name of the last field in the record is specified in the first portion. Auxiliary information for documentation should be included, such as record size, whether fixed or variable length, etc. The auxiliary information starts at the beginning of the comments column.

This entry will be converted to a standard NAME entry before going into the Autocoder processor. Subsequent RNAME entries will give the effect of a SASN to the beginning of the file area, thus allowing a redefinition to occur. As a new file is to be described we get the effect (LASN) of starting storage assignment just above the definition of the longest record in the previous file.

GNAME

The GNAME (group name) class entry permits a continuous section of storage to be given a name. These groups can then be broken down into individual fields. GNAME operates much like that of the NAME entry in Autocoder III. (Note exception below.) Groups of fields may be nested within other groups--there can, however, be no overlapping of these definitions.

PERMITTED			NOT PERMITTED		
NAME1	GNAME	NAME6	NAME1	GNAME	NAME5
NAME2	FLD		NAME2	FLD	
NAME3	GNAME	NAME5	NAME3	GNAME	NAME6
NAME4	FLD		NAME4	FLD	
NAME5	FLD		NAME5		
NAME6	PRE		NAME6		

A GNAME entry used in table 0000 (input-output definition) may not contain a value in the numerical column, since the starting position is indicated with the RNAME entry.

The name of the ending field of a group is specified in the description portion of the sheet. The ending field <u>is</u> included within the group.

FLD

This is used to assign names to data fields in the input-output areas, specify the length, and special characteristics of the data. Unassigned variables are specified in this manner. FLD corresponds to RCD of Autocoder.

FLD appears in the class column of the Name Description sheet. If subsequent entries are also field descriptions, FLD need not be written, as they are considered to be the same as the previous entry.

The length of each field within the record is written in the numeric column of the program sheet. Memory space is allocated equal to the number of positions specified for each field. When a field length exceeds 99 positions, overflow from the two-digit numeric field may extend up to three placed to the left (into the class column) if the class column is blank.

Individual fields within the record should be defined according to the type of field by placing one of the following code characters in the first (or first and second) position(s) of the description portion of the sheet.

- N if the field is an unsigned integer
- \+ if the field is a signed integer
- A if the rightmost character of the field may not always provide left protection for the adjacent signed field
- A+ if the field is a non-numeric field whose rightmost character will always provide left protection for an adjacent signed field

In addition to the aforementioned code characters, the format of all numeric fields which contain a decimal point should be outlined in the operand column with x's immediately following the code character. The position of the decimal point should be properly noted within this field of x's. Another means of specifying decimals may be used alternatively. The format is #+xx.yy where + represents the sign (blank for unsigned), xx the number of integers, a decimal point, and yy the number of decimal positions. Record marks and/or group marks are indicated by placing the characters in the operand column. If these characters are to be considered as terminating some field, all of the terminal characters should be defined on the same line of the sheet.

PG	LINE	NAME	CLASS	NUM.	DESCRIPTION	COMMENTS
01	07	YTDUSE	FLD	12	+XXXXXXXXXX.XX	UNITS OF USE THIS YEAR TO 2 PLACES

reserves an area in memory for a twelve-position signed integer for the year-to-date units of usage. Two decimal places are specified.

PG	LINE	NAME	CLASS	NUM.	DESCRIPTION	COMMENTS
01	08	LASTBILL		6	+XXXX.XX	AMOUNT OF LAST BILL

Although the class column is blank, the entry is understood to be an FLD as the most proximate previous entry is an FLD (0107). Six positions are allocated for this field.

PG	LINE	NAME	CLASS	NUM.	DESCRIPTION	COMMENTS
02	03	USEACCOUN	FLD	7	N	ACCOUNT NUMBER

Seven positions are being reserved for an unsigned account number

PG	LINE	NAME	CLASS	NUM.	DESCRIPTION	COMMENTS
02	04	USENAME	FLD	20	A+	CUSTOMER NAME APPEARS ONLY ON NEW
02	05					AND CANCELLED ACCOUNTS

A twenty-position field is being reserved for the customer's name. The A+ in the operand indicates that the name will always be a non-numeric field whose rightmost character will provide left protection for an adjacent signed field.

PG	LINE	NAME	CLASS	NUM.	DESCRIPTION	COMMENTS
	01	LONGFIELD	FLD	99	A	
	02			5	A	
	03					

Two lines are needed to define a field in excess of 99 positions in length. If the class column is blank, the three rightmost positions may be used in conjunction with the numeric column to define a five-digit record length.

PRE

Used to name, indicate size, and specify values for data fields. These preassigned variables are specified in the same manner as CON in Autocoder. The entry PRE must appear in the class column of the operand sheet for the first line of the constant being defined. All immediately succeeding entries with blanks in the class column will be understood to be PRE entries. Any combination of alphabetic, numeric, special characters and blanks may be specified as a preassigned variable.

The length of the constant is written in the numeric column. The value is placed in the description column. Although a constant may be of any length, only 52 characters can be written on any one line. This is the number of positions on the operand sheet. Consequently, when the actual value written on the program sheet uses the entire line, the highest permissible number in the numeric column is 52. If a number higher than this is written in the numeric column, the system will add zeros to the right of a signed value, and will add blanks to the right on an unsigned value. When the length specified in the numeric column is less than the actual number of characters, the system will shorten the value to the length specified in the numeric column by omitting the right-hand characters.

Record marks and/or group marks are indicated by writing the character(s) in the description column. If these characters are to be considered as terminating some field, all the terminal characters should be defined on the same line.

A numeric field may be signed by writing a plus or minus sign preceding the field. Only numeric constants should be signed. This results in the signing of the last digit. When defining a signed value which contains decimal digits, the decimal point is not included in the count of characters which is written in the numeric column. However, when defining an unsigned value containing decimal places, the decimal point must be included in the count of characters.

The following are some examples of PRE entries.

PG	LINE	NAME	CLASS	NUM.	DESCRIPTION	COMMENTS
04	13		PRE	38		

A 38-position field of blanks is defined.

PG	LINE	NAME	CLASS	NUM.	DESCRIPTION	COMMENTS
07	05	MASTSAVERM	PRE	1	‡	

A record mark is defined.

PG	LINE	NAME	CLASS	NUM.	DESCRIPTION	COMMENTS
07	06	RATE		3	+.875	

Although the class column is blank, the entry is understood to be a PRE as the most proximate previous entry is a PRE (0705). A rate of +.875 per unit is defined as a preassigned factor. As this is signed, the decimal point is not included when specifying the number of characters in the numeric column.

RPT

This is used to define report field formats, also indicating form and size. The entry RPT must appear in the class column of the program sheet when defining a format for a numeric field in a printed report. Actions referencing fields defined by RPT have the ability to produce the coding necessary to place a numeric field into the desired format containing decimal places, commas, sign indicators, floating or fixed dollar sign and asterisk protection to the first significant digit to the right of a fixed dollar sign.

The numeric column of the RPT entry contains the number of positions required by the entire report format field. In determining the size of an RPT field, allowance must be made for the maximum number of numeric digits, the dollar sign, commas, decimal points and the number of characters (1 or 2) used for sign indications.

The description of an RPT entry must appear in the following order:

1. When a dollar sign is desired, the first character of the field must be a dollar sign which is followed by a representation of the field as it is to appear in the report. X's

are used to indicate numeric characters. If the printing of insignificant zeros is desired, Z's must be used instead of X's in the affected positions. Z's must be continuous in the RPT format field starting from the low-order position. If any commas are used, they must appear in the normal manner, i. e., the fourth position to the left of a decimal point or sign indicator and thereafter every three additional positions to the left. Only one decimal point may be used. The positions allocated for sign indication must be indicated as blanks preceding the first lozenge.

2. The dollar sign will always appear in the position assigned regardless of the number of digits printed unless a floating dollar sign is specified. If a floating dollar sign is specified, any dollar sign printed will appear immediately to the left of the high-order digit. In order to specify a floating dollar sign, the character $ must be written between the first and second lozenges.

3. The spaces between the second and third, third and fourth, and fourth and fifth lozenges are used for negative, zero and positive sign indication respectively. Any of the following symbols may be used for sign indication:

Length	Symbol
1	Sign indication omitted (lozenges may be placed back-to-back, but one blank position is generated.)
1	-
2	CR
2	DB
1	*
2	Blank followed by -
2	Blank followed by *
2	**
1	+

If no sign indication is specified, the system automatically will provide a - for minus indication and no sign indication for plus or zero. When specifying the length in this instance, the automatic minus indication must be included. If no dollar sign control is desired, but sign indication is to be used, all five lozenges must be specified, the first two lozenges being placed back-to-back.

The following are some examples of RPT entries.

PG	LINE	NAME	CLASS	NUM.	DESCRIPTION	COMMENTS
04	07	BILLUSE	RPT	7	XXXX.Z	CURRENT USE

A seven-position field is specified with one decimal position. Insignificant zeros will be printed in the low-order position. No sign indication is specified; therefore, it is assumed that the automatic sign indication provided by Autocoder is desired. The "7" in the numeric column includes the automatic - for minus indication.

PG	LINE	NAME	CLASS	NUM.	DESCRIPTION	COMMENTS
05	13	DELUSE	RPT	11	XXXXXXXZ.Z	UNPAID USE

An eleven-position field is specified with one decimal position. Insignificant zeros will be printed in the two low-order positions. Automatic sign indication will be generated as in the previous example.

PG	LINE	NAME	CLASS	NUM.	DESCRIPTION	COMMENTS
04	14	BILLAMOUNT	RPT	10	$X.XXX.ZZ ⌑$⌑	AMOUNT BILLED--FLOATING DOLLAR SIGN

A ten-position monetary field is specified. The ten positions include all the positions in the operand up to the first lozenge, i.e., the dollar sign, four integers, two decimal places, one comma, one decimal point and one position for sign indication. Insignificant zeros will be printed in the two decimal places. A floating dollar sign is specified. As in the previous examples, no special sign indication is specified; therefore automatic sign indication will be provided by the system. Note that it is permissible in this instance to omit the lozenges for sign indication.

PG	LINE	NAME	CLASS	NUM.	DESCRIPTION	COMMENTS
			RPT	12	$XX,XZZ.ZZ ⌑⌑⌑C	R⌑⌑⌑⌑

Insignificant zeros will print in the cents and two low-order dollar positions of this monetary field. A fixed dollar sign with asterisk protection is specified along with a CR symbol for negative indication, an * for zero amounts and nothing for positive indication.

BITCD

This is used to permit the naming of various bits in a character. Later they can be tested for ON or OFF in the condition area, and can be put ON or OFF by the MOVE and SET commands.

The BITCD definition permits the user to associate a meaning with the presence or absence (the ON or OFF state) of individual bits. The BITCD header signals the system that a single character position is to be set aside for a set of up to four codes. The 1, 2, 4 and A bits of that character location may be used as ON-OFF switches to indicate the presence or absence of some condition. The entries which follow the BITCD should be blank in the class column. The numeric column is used to specify the bit (1, 2, 4 or A) with which a particular condition is to be associated. The condition itself is named in the name column.

Specifying more than one condition at a time is possible as more than one bit may be present within a character at one time.

The following is an example of a BITCD entry which could be used in a payroll application.

PG	LINE	NAME	CLASS	NUM.	DESCRIPTION	COMMENTS
		DEDUCT	BITCD			
		USBONDS		1		
		HOSPITAL		2		
		STOCK		4		

Since any number of three bits of the memory position occupied by this BITCD entry may be on, payroll deductions may be made for none, any or all of the three options available.

TABLE 0001 - Core Files

Working storage files (core) are described immediately following the table header information identifying this section. The files and records described here are never read into or written out of the machine directly; they are for intermediate values. A number of possible uses are seen: when more than a single input record must be available they can be moved into the intermediate area, when various output records are being fabricated at one time for later movement to an output file, e. g., multiple total and heading lines, and for multiple definitions of an area when operating on variable length records. The RNAME entry is used to reinitialize back to the beginning of the area when a new format is to be described.

The rules for describing these files are precisely the same as for input/output files, except that an IOCS File Table is not prepared. Information cannot be read into or written out of I/O devices directly from these areas, except that they can be the second operand of a GET or PUT action. In this case the data is moved to/from the I/O area after or before the I/O action takes place. When more than a single format is desired for a core file, the RNAME entry will permit the redefinition. If multiple record areas are to be defined, even if they are eventually to be outputted on the same file, separate FILE entries should be made each with a single record described.

TABLE 0002 - Miscellaneous, Variable

Free or unassociated variables of one kind or another are described behind this table header. Eight classes of entries are permitted.

GNAME
FLD
RPT
PRE
BITCD
ALTSW
ADCON
POINT

Except when grouped with the GNAME class, there is no physical connection assumed among any of the entries. No FILE or RNAME entry is allowed in Table 0002. This section is for temporary storage fields, program bit switches, preassigned variables, alteration switch specification, address constants, and index pointers.

GNAME -

Used when multiple items are to be considered as an entity. Example:

CURDATE	GNAME	06	YEAR
MONTH	FLD	02	N
DAY		02	N
YEAR		02	N

Any combination of RPT, FLD, PRE, BITCD, ADCON and POINT can be grouped and treated as an entity by this entry.

FLD -

To define an unassigned variable, e. g., TEMP1, ERASE1, PREVMANNO. Defined in same manner as described above.

PRE -

To define a preassigned variable, i. e., starting with an initial value, e. g., COUNTER1, INITIALVAL, YTDTOTAL. Defined in same manner as described above.

RPT -

To define a variable and associate particular editing symbols. Defined in same manner as described above.

BITCD -

To set up a one-character variable and then name the 1, 2, 4, and/or A bits such that they can be turned ON or OFF by the program. They can also be interrogated by the program to see if they are ON or OFF.

DEDUCTIONS	BITCD		
BONDS		1	
STOCK		2	
BLUECROSS		4	
INSURANCE		A	

ALTSW -

This permits the naming of an alteration switch. The designation of the proper alteration switch, A for 0911, B for 0912... F for 0916, is placed in the numeric column.

QUARTERLY | ALTSW | C |

ADCON -

Address constants will have limited use in the 7080 DTS. Three primary uses are seen (1) by closed Autocoder pseudo tables, (2) for use in developing initial base addresses when using index pointers for operation on variable length input or output files, and (3) the pre-named Address Registers R0 - R9 used for effecting address modification. ADCON 4, 5, 6 are not allowed in the Decision Table Name Description.

POINT -

The POINT (index pointer) entry is used to identify a 6-position field to be used in conjunction with the MOVEV command when operating positionally on data. Normal arithmetic is used to modify the field in preparation for use. At the time of use, the system will convert this to a proper 7080 address. It is always considered to be an integer value of length 6. The field may be either signed or unsigned (signed is more efficient).

The name of an index pointer may be from one to five characters in length. The numeric column is not used.

PNTR1 | POINT | | $\begin{Bmatrix} \text{HI} \\ \text{SER} \end{Bmatrix}$

The operand HI indicates that the POINT will be initialized to a high speed address, modified by a number divisible by 5, and that the number of characters to be moved via the POINT will always be divisible by 5. The operand must be a RECORD name (RNAME) if HI is specified. If the operand is blank, SER (serial) will be assumed. POINT generates a 6 position signed constant of zeros. An index pointer can only be initialized with the SET... EQ... command.

TABLE 0003 - Miscellaneous, Fixed

This table includes items that do not have changing values--either constant values or expressions that have a value only when considered. Since they have this characteristic (like the program tables) they can, if desired, be stored with the tables as part of the program using them in some external medium to be called for when needed. Included are:

- Expressions
  - Arithmetic
  - Symbolic
- Constants
- Value Lists
- Reference Lists

EXPRESSIONS -

Two types of expressions are used--arithmetic to operate on numeric values, and symbolic to operate on strings of symbols.

Arithmetic -

Arithmetic expressions are defined in much the same manner as for the MATH macro in the Autocoder III system. The following operators are available:

+	addition
-	subtraction
*	multiplication
/	division

A single level of parentheses is permitted to alter the normal precedence of operations, which calls for parenthesized portions of an expression first, with * and / always taking precedence over + and -. A left-to-right order of operation then is assumed.

An arithmetic expression is named in the normal manner. It is this name that is referenced in the program, thus calling for the proper computation.

The class entry is specified as MEXP for mathematical expression.

The numeric column is used to indicate the type of error protection desired by the user. Three codes are available:

1.	⊙	Overflow	Overflow occurs when the developed value is larger than the defined result field. A transfer to the open decision table specified at the end of the expression takes place when overflow occurs. If no name is specified, the table given in the last expression will be used.
2.	T	Truncate	The result will be truncated rather than rounded (the normal procedure).
3.	⊙T	Overflow/Truncate	The value is truncated and transfer is made to the table specified if the value is too large.

The halt protection code (H) is not permitted, also the chaining code (X) is not permitted.

The description field is used to specify the format of the result, the expression itself, and if desired, the transfer point for the overflow protection option.

The format of the final result is indicated by #+xx. yy, starting in the first position (column 31) of the description field. The + indicates that the result should be signed (a blank will indicate an unsigned result); xx, the number of integers; a period; yy the number of decimal positions in the result; a comma.

The format specification is followed by the expression to define the computation. The operators are +, -, *, /; the delimiters left parenthesis, right parenthesis, and @ (if overflow protection is desired); the operands literals, constants, preassigned fields, variable fields, and previously defined arithmetic expression names; and special intermediate value format indicators (a comma followed by an x for each integer, a period, and an x for each decimal--, xxxx. xx).

Numeric literals should not be enclosed in literal indicators (#), and may not be given a sign--they are all positive values. Also, the unary operator - may not be used, i. e., the construction

FIELD A + - FIELD B is not allowed. The format indicator after a parenthesized expression or single variable will cause that format to be used before the next operation takes place. Blanks should precede and follow each operator (+, -, *, /) in the expression.

If overflow protection is specified in the numeric column (⊙ or ⊙T) the expression is terminated by the @ symbol and followed by the name of the open decision table to which control is sent on overflow. Otherwise the @ symbol is not used. In no case is the result field specified here as in Autocoder (the tag is specified in the name field on the operand description sheet). If overflow is indicated and no address is given, the last specified address is used.

Note that the left parenthesis, right parenthesis, and plus sign are used in writing the expression; these are keypunched as %, ⌑ , and &, respectively.

Examples:

FIELDA	MEXP		# + 04. 02, FIELDB + FIELDC * FIELDD
ABLE	MEXP	⊙	# 02. 04, B + (C * 1. 5), xx. xxxx - (I-J) @ OVFTABLE
X	MEXP	T	# + 06. 02, (A - B * C) + D / (F - G * H) - (I + J)

NOTE: There is no permanent value of an expression; every time it is referred to it is recalculated. The value can be saved of course by moving it to another field. A previously defined expression can be used as an operand in an expression.

All continuation cards (up to 9 permitted) must be blank in columns 6 through 22 and begin in column 23. Comments on any card of a MEXP will be lost.

Symbolic -

Only a single symbolic operator is available, the JOIN operator. This operator allows strings of characters to be joined together from discontinuous points in storage and then treated as a single value.

Variable names, constant names, symbolic expression names, and literals may be operands in a symbolic expression. The operands are separated by commas and as a group are enclosed in parentheses. Remember that (will be keypunched as % and) as ⌑ .

Example:

SORTKEY	SEXP	14	JOIN (MAJOR, INTERMED, MINOR)
ERRORMESG	SEXP	22	JOIN ('TAPE ERROR ON UNIT', TAPEUNIT)
IDENTNO	SEXP	15	JOIN (DEPTNO, ' - ', DIVND, ' / ', MANNO)

In this last example if the current values of

DEPTNO = 726
DIVNO = 14
MANNO = 123456

the resulting value of the expression would be

726 - 14/123456

The numeric column is used to specify the total length of the resulting value. The quote symbol (') is used to delimit alphameric literal values. This is keypunched as the number sign (#).

It can be seen that an entire print line can be created in this manner by listing the appropriate carriage control character, blanks, constants, and variables, e. g. ,

JOIN ('. . . ', IDNUMBER, ' ', NAME, ' ', ADDRESS, '***'. . . .)

A symbolic expression can call upon another <u>previously</u> defined symbolic expression by including its name in the list of names and literal values. It cannot call upon itself; this causes an error condition. Up to 9 continuation cards are permitted; they must be blank in columns 6 through 22.

CONSTANTS -

Permanent constants that are never or very seldom changed, e.g., 3.1416, are specified in this section. The format and manner of specification is the same as Autocoder III except that the class is PCON and no value can be moved to the field in the program, i.e., they are 'read only' constants.

```
PI      PCON    06    +3.14159
RATE    PCON    03    + .875
```

Numeric signed constants are indicated by placing the sign first (+ or -) followed by the value. An assumed decimal point is written in the proper position. No point is needed for integer values. The numeric column gives the length of the value; its sign and point indicator are not counted in determining the total size.

Alphameric constants are also written in the description area, following the length specification in the numeric column.

VALUE LIST -

A value list consists of a list of constant values that can be used in the condition section of a program table. For example, in a condition it might be desirable to see if the current value of an input item, e.g., department number, is contained in a list of many department numbers which might require special action, e.g., 214, 284, 319, 727, 914.

NAME - A name is assigned to this list--this name is then used as an operand in the program to refer to the entire list.

CLASS - The class code is VLIST (Value List)

NUM - The number of positions of each value in the list--all values must be of the same length (blanks or zeros permitted front or back).

DESCRIPTION - The values are listed one per line starting in the first position of this area. Only constant values can be specified--no names of values (tags). No duplicate values should appear. The values need not be enclosed within special symbols. The values may be either numeric or alphameric.

Example:

SABRECITY	VLIST	3	ATL
			MIA
			OHR
			SFO
			LGA
			IDL

The values in the list need be in no particular order; when interrogating the list, a serial search will be assumed.

REFERENCE LIST -

A reference list provides a set of constant arguments and values (functions) to be specified by the user. It is then possible to determine if a particular argument is in the table (in a condition) and use the value as the operand in an action.

NAME - The name of the reference list

CLASS - RLIST (reference list)

NUM - Blank

DESCRIPTION -

Header Line - Three parameters are specified in the first line. A two-position size of the argument, a comma, a two-position size of the value (function), a comma, the name of the field where the value is to be placed if the argument is found when searching the table.

Subsequent Lines - The name, class, and numerical columns must be blank. Beginning in the first position of the description field (Column 23) and extending through the comments field, the argument and function are written separated by a comma. The argument (characters prior to the comma) and function must occupy exactly the same number of positions as specified in the parameter of the reference table definition. The first position of the argument may not be defined as a plus (+) or minus (-). An entry definition may not be continued on the next card.

If there are many arguments associated with one value and if the arguments are numeric and in ascending order, a range argument may be used which can save the writing of many entries. This is expressed by giving the low limit and the high limit separated by a dash (02-05, MIDDLE ATLANTIC). The arguments for this type of entry must be pure numeric. Care should be taken to write this entry correctly. For example, 0100-1000 will generate 901 entries.

```
STATELIST  RLIST  02, 04, STATE
                  01, CONN
                  02, MASS
                  04, TENN
                  07, KANS
```

Arguments or function values can be either alphameric or pure numeric (note restriction in use of range above). The arguments may or may not be in ascending sequence.

NOTE: A maximum of 25 RLISTs is allowed in a single program.

IBM 7080

DECISION TABLE NAME DESCRIPTION SHEET

PROGRAM ____________

DATE ____________

PREPARED BY ____________

PAGE ____________

NEXT PAGE ____________

PG	LINE	CC	TYP			IDENT	TABLE NAME	
	0 0 0	0				T A B L E		
1 2	3 5	6	7	8 11	12 15	16 20	21 30	31 74

IDENT. 75 80

PG 1 2	LINE 3 5	NAME 6 15	CLASS 16 20	NUM. 21 22	DESCRIPTION 23 38	COMMENTS 39 74
	01					
	02					
	03					
	04					
	05					
	06					
	07					
	08					
	09					
	10					
	11					
	12					
	13					
	14					
	15					
	16					
	17					
	18					
	19					
	20					
	21					
	22					
	23					
	24					
	25					

Name Description Header

1 - 5	Page and Line are filled out in the normal fashion. All cards entering the system must be in ascending sequence.
6	Always zero.
7	Type for operand description should be the letter D (description).

Since this same form and header are used for Report and Autocoder pseudo tables, a code is used to identify each. R is used for report and A is used for Autocoder.

8 - 11	Blank.
12 - 15	Table Number is specified in this position. Only four numbers are possible:

0000 Input/output File Description
0001 Working storage files
0002 Miscellaneous, Variable
0003 Miscellaneous, Fixed

Each table gets only a single header card; if a particular operand description table is not used, no header is specified. A maximum of four name description headings may appear (ignoring report or Autocoder pseudo tables which are really part of the procedure description).

16 - 20	Identification always TABLE. NOTE: TABLE cannot be used in these positions except to identify an operand or procedure description table.
21 - 30	Table Name may be specified as documentation but is never referenced from another part of the system.
31 - 74	Blank or used for remarks.
74 - 80	IDENT is used to identify the cards as part of a particular program, is never referenced by the program, i. e., normal Autocoder use.

REPORTS

It is of course possible to prepare reports using regular 7080 decision tables, but since many of the functions are automated in Report Writer the user may wish to call upon this facility. Each different report that he wishes is specified in a separate pseudo table starting with a table header. The report is named in the header; this name is used as the operand of the DO command when the report routine is to be executed. Table type is specified as 'R' (report).

The report is then described using the operand description sheets (same format as Autocoder) in the same manner described in the Report/File Writing manual of the 7058 Processor series.

Three other special rules apply: (1) The transfer-out addresses (before or after control breaks) must reference a closed decision table only. This table can perform all the processing at the control break or call for other tables or Autocoder sections on a DO basis. At the completion of this excursion, control returns to the report routine and then back to the original calling table. (2) A MODE REPORT entry must precede Report Writer statements and a MODE AUTOCODER entry must follow the last statement. Both entries must be included in the 'R' table. (3) The DREPT statement, the first statement of a Report section, must be tagged with the report Table Name. The programmer will DO table name to execute a Report Writer table.

AUTOCODER

Pseudo tables can also be written in the Autocoder language. Again, a table header must precede the routine (table type is A for Autocoder). The procedure written in Autocoder must be a closed routine, i.e., it may not transfer out to other tables or pseudo tables. The program refers to the elements of data described in the operand description tables (0000 - 0003), rather than defining data in the Autocoder section. The Autocoder TABLE macro in particular must be avoided. No restrictions (except transfers noted above) are placed upon the instructions written in the Autocoder sections. The programmer need only code the routine itself without entry and exit points. Entry will be at the first command, exit will occur after the last command. (See Reference Manual--Programming the IBM 705 Using the Autocoder III System--No. C28-6057.)

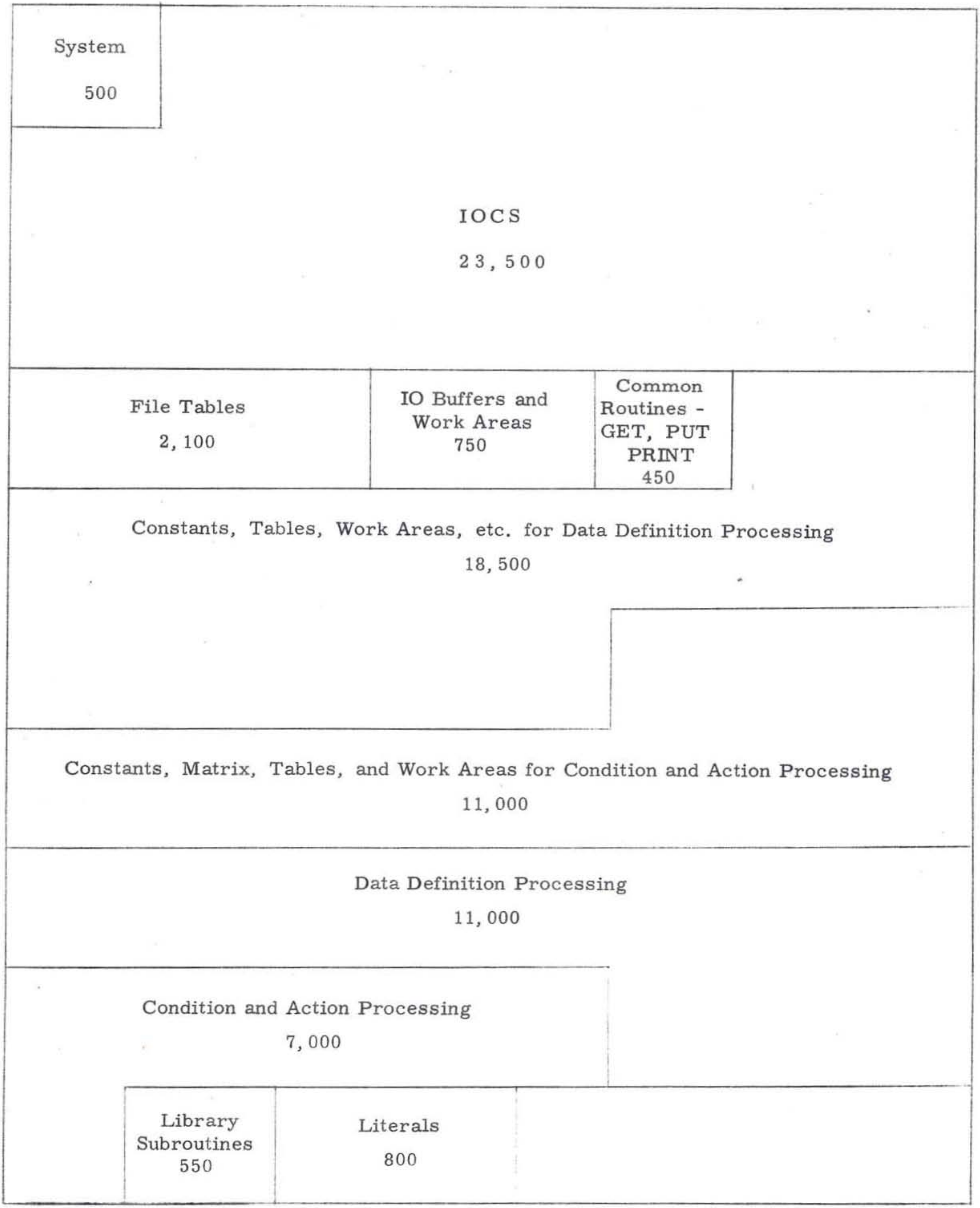
APPENDIX

Character Set

The character set for source programs (reference set) is shown below with the appropriate card codes. The printing characters which differ from the reference set are shown in the second column (¤ , &, %, #). This chart also shows the machine collating sequence. Characters that may appear internally (although not part of the external character set) are shown in their collating position and marked with an asterisk (‡ , $\overset{+}{0}$, $\bar{0}$, †). It is expected that the key punch operator will substitute the ¤ , &, %, # for), +, (, '(quote) as she punches the cards (the programmer may also use the printing set if he desires).

Reference Set	Printing Set (where different)	Card Code	Reference Set	Card Code
Blank				
.		12-8-3	M	11-4
)	¤	12-8-4	N	11-5
‡		*	O	11-6
+	&	12	P	11-7
$		11-8-3	Q	11-8
*		11-8-4	R	11-9
-		11	†	*
/		0-1	S	0-2
,		0-8-3	T	0-3
(	%	0-8-4	U	0-4
'	#	8-3	V	0-5
@		8-4	W	0-6
$\overset{+}{0}$		*	X	0-7
A		12-1	Y	0-8
B		12-2	Z	0-9
C		12-3	0	0
D		12-4	1	1
E		12-5	2	2
F		12-6	3	3
G		12-7	4	4
H		12-8	5	5
I		12-9	6	6
$\bar{0}$		*	7	7
J		11-1	8	8
K		11-2	9	9
L		11-3		

7080 DTS MEMORY MAP



Notes on Pre-processor Requirements

The 7080 Tabular Programming Pre-processor is written for an 80K, 2 channel 7080, with a typewriter. It operates in non-stop mode using the 7080 IOCS package. It may be run on a 160K, 2 or 4 channel 7080 with no modification.

Input

Input is unblocked 80 character records on tape. The first Tabular Program on tape must be preceded by a run control card as follows:

col	16-20	ASMBL
col	23-29	TABULAR

This card may be preceded by a DATE and/or MODEL control card as described in the 7058 Processor specifications. Each Tabular program on tape must be separated by a card having END punched in col 16-18. The final program on tape must be ended by a card as follows:

col	6-10	FINAL
col	16-18	END

Input is limited to a single reel. A tape mark on the input tape is considered to be FINAL END.

Output Program

The processed Tabular Program will be unblocked 80 character records on tape, and will be acceptable to the 7058 Processor. Since the 7058 processor is unable to accept multi-reel input, no program may extend over one reel.

If end of reel on output does occur in the 7080 DTS the following will occur:

1. Input will be rewound.

2. Output will be tape-marked and re-wound.

3. A message will be put on the message tape indicating that the current program will be re-processed.

4. A message to the operator will be typed. The operator must mount a new output reel and press key 252.

5. 7080 DTS will then search the input tape until the first card of the program being processed is found. At this point, an ASMBL card will be generated and placed on the new output tape, and processing of the program will begin again.

Output Messages

Output Messages will be 120 character records blocked 5, on tape. The format will include the 7080 DTS Pre-processor page and line number and the input page and line, tag, and operations of the entry to which the error message applies, a critical/non-critical indicator, and the text of the message.

Messages for each program will begin on a new page, and each page will be headed by a line containing the program identification and date of processing.

Messages may be printed on a 720 or 1403 printer under program control.

Machine Setup

Non-stop switch	on	
Alteration switches		
911	on	for testing the processor
	off	at all other times
912	off	
913	on	if the operator desires the processor to search for END cards and give the option of processing the program following a found END card or continuing the search.
	off	otherwise
914	off	
915	off	
916	off	

Tapes

2001	Tabular program input Hi or Low Density, Model II or IV tapes.
2101	Tabular program output will be Hi Density Model II or IV tapes.
2103	Message output will be Hi Density, Model II or IV tapes.

Tape assignments may be changed by patching the Tape Table entries which begin at actual location 500.

No checkpoints will be taken, and no records will be dumped. If an uncorrectable read or write error occurs, new tapes must be mounted and processing must be started over.

When a program has been completely processed, (an END card has been found) a message will be typed containing the program ident., and indication of what kind of errors were found (major, minor, or none), and giving the operator the option of processing the next program, or going to end of job.

When a FINAL END card ends a program, the same message is typed, but no option is given. End of job follows automatically.

Typewriter Messages - Pre-processor

The following typewritten messages may appear during the compilation of a 7080 Tabular Program.

7080 TP

appears at the beginning of a compilation run only.

XXXXXX, MO-DA-YR

where XXXXXX is the identification of the Tabular Program to be compiled. Typed at the beginning of compilation for each program.

X-----X. - IMPROPER CONTROL CARD. -2-ACCEPT, -3-READ NEXT

where X-----X is the PGLN, TAG, OPERATOR of the control card in question. May appear at the beginning of a compilation run only. -2- will assume an ASMBL TABULAR control card, output an ASMBL ORIGINAL card, and begin processing. -3- reads the next card.

ASMBL X-----X IMPROPER. -2-ACCEPT,-3-QUIT

an ASMBL card with operand X-----X has been found at the beginning of a compilation run. -2- will assume TABULAR, output card as is, and begin processing. -3- will go to end of job halt.

XXXXXX IS NEXT. -2-SEARCH, -3-PROCESS

where XXXXXX is program identification. Will appear when 913 is on at beginning of a run. An END card has been found and XXXXXX is the next program on tape. -2- continues the search -3- generates an output. ASMBL ORIGINAL control card and begins compilation.

XXXXXX COMPLETE, YYY ERRORS. -2-NEXT ASSY, -3-EOJ

Program XXXXXX has been compiled, and YYY is the class of errors found. Will be NO if no errors found, MIN if Minor errors found, or MAJ is a critical error found. The option section of the message will appear if an END card, not FINAL, indicated the end of the current program. If FINAL END encountered, end of job is automatic, and the message will be truncated at the period.

Typewriter Messages - Listing Program

XXXXXXX NEXT. -2-LIST, -3-BYPASS
XXXXXXX NOT LISTED.
XXXXXXX LISTING COMPLETE.

Message Tape Entries

I. Operand Description Section

A. Message Format

TP. PGLN		PGLN	NAME	CLASS	CRIT	MESSAGE TEXT	IDENT

Where:

TP. PGLN is the Page/Line Number assigned to the first output entry of this item by 7080 DTS.

PGLN, NAME, CLASS are the first 20 positions of the input entry.

CRIT is blank if this error is minor; is *** if this error is considered a major error. Major errors are those errors which are difficult or impossible to correct at output (Autocoder III) level.

IDENT is the input program identification.

B. Message Texts:

DUPLICATE TAGS - EXPRESSION OR LIST

The tag of this item is the same as the tag of a previously defined expression, list, index pointer, or is a state relation name.

EXPRESSION REFERS TO ITS OWN TAG.

An arithmetic or symbolic contains its own tag as one of its numbers.

EXPRES. NOT TAGGED OR BLANK OPND-PROCEDURE TABLE
(CRITICAL)

An expression, list, or index pointer is not tagged, and cannot be referenced. See procedure table error messages for explanation of second half of message.

FIRST RNAME IN FILE NOT TAGGED-WILL TAG

The first RNAME entry in a File which contains 2 or more RNAME definitions is not tagged. 7080 DTS will create a tage for this RNAME and subsequent RNAMES within this file will be LASNed to this tag. This LASN will not be effective in the Autocoder assembly.

IMPROPER TYPE FOR CURRENT TAB NO.

This item is a valid class type for an Operand Description, but is not allowable within the present TABLE number. Entry is processed normally.

L. H. BRKT NOT LOC. -ASSUME EXP BEG. COL28.

The left hand bracket of a symbolic expression does not appear in col. 28 or 27 of the input entry. De-coding of this expression will begin at column 28 of the input record.

LES RNAMES THAN DEFINED-LAST FILE

The numeric column of the previous FILE entry contained a number greater than the number of RNAME entries processed before reaching the present FILE entry. Processing continues normally.

MORE RNAMES DEFINED THAN FILE CALLED FOR

The current RNAME entry is over the number of RNAMEs which the previous FILE contained in its numeric column.

NO EXP TYPE-ASSUME JOIN

The current symbolic expression has an invalid or blank expression type in col. 23-26. The JOIN type is assumed.

OUT OF SEQUENCE

The Page/Line of the current entry is lower than that of the previous entry.

OVER 9 CONTINUATION CARDS FOR EXP. -IGNORED (CRITICAL)

The present entry is the 10th continuation card of an Arithmetic or Symbolic expression. It will be re-coded as a TITLE entry and the expression will end at the end of card 9.

OVER 25 RLISTS - TREATED AS VLIST

The current RLIST entry is number 26 or above. No movement of a function will occur as the result of its use in a condition statement.

TAG OF POINT OVER 5 CHAR. LONG

The current index pointer name is 6 or more characters in length. It will be processed normally.

TBL. TYPE INVALID TAB. NO. -ASSUME XXXXXX

The current TABLE header is type "D" (data) but has a Table Number greater than 0003 (only the numeric portion of a Table number is used in this test). An assumption of Table number/type will be made on the basis of the operator of the following entry. If ORDER, a procedure Table is assumed. If FILE, Table 0000 is assumed. If neither, Table 0003 is assumed.

UNKNOWN OPERATOR TYPE

The operator of the current entry is not in the list of acceptable operand description classes. The entry will be put out as is.

***EOF OUTPUT -PROCESSING RE-STARTED**

End of file on the output tape has occurred. The current program will be recompiled from the beginning. Error messages up to this point will be repeated on the next page of error messages.

X------X QUESTIONABLE OPERAND FOR EXPRESSION

The current arithmetic or symbolic expression references an index pointer name, list name, or state relation. May be improper usage.

INVALID, TAB. NO.-USING NAME ONLY

The Table Number of an Autocoder (A) Table is not pure numeric or is less than 0004. All procedure Table must have pure numeric Table Numbers greater than 0003. The Table Name will be used to tag the first generated instruction of the Table. DO statements referencing Table number will generate incorrectly.

II. Procedure Section

A. Message Format

TP. PGLN	TAB. NO.	TAB. NAME	CRIT	MESSAGE TEXT	IDENT

Where:

TP. PGLN is the Page/Line number assigned to this output entry by 7080 DTS.

TAB. NO. is the current Procedure Table number. TAB. NAME is the current Procedure Table name. CRIT is blank if this error is minor; is *** if this error is considered a major error.

MESSAGE TEXT is a description of the error.

IDENT is the input program identification.

B. Message Texts:

X------X Y------Y BOTH LISTS-1st IS ARG

Both operands of a condition statement are list names. The first operand will be processed as the name of the argument, the second operand as a list name. This is improper usage, and should be corrected prior to Autocoder assembly.

X------X Y------Y BOTH STATES-1st IS TAG

X------X, Y------Y are operands of the current condition statement and are both state relation names. Operand 1 will be processed as a field name.

X------X- EXPRESSION IMPROPER IN THIS OPERAND

X------X is an arithmetic or symbolic expression name and is the resultant or receiving operand of a MOVE or SET action. No linkage to the expression routine will occur; and this operand will be processed as the name of a signed field.

XX - IMPROPER OPERATOR TYPE FOR CONDITION

XX is the operator of the condition statement being processed, and is invalid. The operator is assumed to be EQ, and processing continues.

X------X- IMPROPER STATE FOR ACTION AREA

X------X is an operand of the action statement currently being processed. It is a state relation, but is not ZERO, BLANK, ON, OFF. These are the only state relations allowed in an action statement. X------X will be processed as a field name.

$$\begin{matrix}\text{SET} \\ \text{MOVE}\end{matrix} \left\{ \text{XX-INVALID OP TYPE-ASSUMING} \right\} \begin{matrix}\text{SET EQ} \\ \text{MOVE TO}\end{matrix}$$

XX is the operator of a SET or MOVE action and is not EQ, +, -, /, *. XX is assumed to be EQ.

X------X - LIST NAME INVALID IN ACTION AREA

X------X is an operand of the action statement currently being processed. List names are not allowable in the action area.

MOVEV X------X Y------Y NO POINT, DID MOVE (Critical)

Neither operand of this MOVEV action statement is an index pointer name and neither operand contained a comma. The action was assumed to be MOVE and processed accordingly.

X------X - POINT NAME UNDEFINED - ASSUME SER.

X------X is the operand of the action statement being processed, and contains a comma. The name to the left of the comma is not an index pointer name, but will be assumed to be the name of a

SER index pointer.

X------X - STATE IMPROPER IN THIS OPERAND

X------X is the operand of the current action statement. It is a state relation name and is improperly used. It will be processed as a field name.

X------X - UNKNOWN ACTION - OPERATOR TYPE (Critical)

X------X is the action/op of the current action statement. It is not a member of the valid action/op list. A NOP, tagged with the tag of this action, and with an operand of this action's operand 1,will be generated.

EXPRES. NOT TAGGED OR BLANK OPND-PROCEDURE TABLE (Critical)

One of the operands of the current procedure table statement being processed is blank. Improper coding will be generated as a result.

X------X TABLE VS STATE-ERROR-NO TABLE SERCH

The present condition statement related a List name and a state name. This is improper usage. The List name will be processed as a field name.

CHECK. NO ELSE RULE OR ERROR TABLE NOTED

Possible error, programmer should make sure all possibilities have been considered.

INVALID. RECORDS BYPASSED TO NEXT TABLE

Unknown record type following actions.

RECORDS MISSING OR OUT OF SEQ - TAB BYPASSED

ZERO ORDER. RECORDS BYPASSED TO NEXT TABLE

ZERO NO. RULES. TABLE BYPASSED.

INVALID ACTIONS. RECS BYPASSED TO NEXT TABLE

Probably no actions for this row.

CHECK. NEXT TABLE IS BLANK.

Possible error, programmer should check to assure a GO TO for each rule.

<u>Listing Program Messages (printed after heading 1)</u>

RECORDS MISSING OR OUT OF SEQUENCE. TABLE BYPASSED.

ERROR. NO. OF RULES IS ZERO. PRINTED AS 6.

ERROR. HAVE ZERO ORDER. RE-ORDER DISCONTINUED.

<u>Notes on Pre-processing Method</u>

The 7080 DTS pre-processor (compiler) converts from programs written in decision table form to acceptable Autocoder III form.

Operand Description --

The compiler makes almost a direct conversion to Autocoder, the major exceptions being that expressions are compiled as part of the data description and are set up to operate on a closed sub-routine basis. IOCS entries are passed without change; FILE is changed to TITLE, RNAME to NAME, GNAME to NAME, FLD to RCD, PRE to CON, RPT is unchanged, VLIST and RLIST to TABLE, MEXP to MATH, SEXP to a special macro GATHR, PCON to CON, BITCD unchanged, ALTSW unchanged, and POINT to ADCON. Address Registers X0 through X9 are changed to 6 digit signed ADCONS.

Procedure Description --

The condition and action statements are converted to Autocoder macro instructions - a number of new macros are used.

<u>Condition area</u> - The entire condition area is in core storage at one time. A slightly different approach is used in scanning depending upon whether the row is in limited or extended entry. Each limited entry test is made before going into the entry portion - a bit is turned "on" if the test is met, i.e., limited entry conditions are pretested, extended entry conditions are not. If this same test must be made more than once, the bit test is very rapid,

thus speeding up the process of finding a "satisfied" rule.

The condition area is then scanned to eliminate horizontal redundancies (marked * in examples below); in addition the reverse tests are eliminated if there are no tests to be made above it in the same rule (marked by a ¤). The rule entries so marked are ignored when the final testing network is compiled. Rearrangement of condition rows and decision rules can greatly affect the number of test to be made in the object programs.

The not pertinent (blank entries) should be moved toward the bottom for space efficiency and often for time efficiency. Other things being equal, the most discriminating (most likely to fail) should be toward the top. Rules should be arranged such that the high frequency rules are at the left, since object testing starts at rule 1. Note examples below, tests are made only when a Y or N (in limited entry) remains.

Blank entries produce the same efficiency at far left <u>or</u> far right if at bottom.

Examples:

Y	Y	Y	produces	Y	*	*
Y	N	N		Y	¤	*
	Y	N			Y	¤
		Y				Y

Y	Y	Y		Y	*	*
N	N	Y		N	*	¤
N	Y			N	¤	
Y				Y		

But blank entries intermingled generate more tests.

Y	Y	Y		Y	*	*
	Y	N			Y	¤
Y	N	N		Y	N	N
		Y				Y

Blank entries at the top (either left or right) give the worst possible arrangement.

		Y		No redundancies
	Y	N		
Y	N	N		
Y	Y	Y		

The rule to remember is: the row above must be *before a or * can replace the test. *'s are assumed to be before the first row. Therefore when table forms a tree, blank entries should be at the bottom.

Action area - Each action appears only once in the compiled program (unless the user writes the same action more than once). During compilation a control word is set up for the actions of each rule (actually Bit switches), "on" indicates "execute" it, off indicates "skip" it. The proper control word is set up when the conditions of a particular rule is satisfied, then the Bit corresponding to each action is tested in turn - if "on" execute, if "off" go to the next Bit and test, and so on for each assigned action in the table. For example:

	Rule 1	Rule 2	Rule 3
Action 1	X	X	X
Action 2	X	X	
Action 3	X		X
Action 4		X	
Action 5	X		X
Action 6	X	X	X

Rule 3 would have a control word -ON-OFF-ON-OFF-ON-ON, and Actions 1, 3, 5, and 6 would be executed. Rule 1 would have actions 1, 2, 3, 5, 6 executed. If long sequences are to be executed for a number of rules they might be written as a separate "closed" unconditional table and executed by a DO Table name action command.