GRAO

UNIFIED SYSTEMS APPROACH

CLEARINGHOUSE REPORT

TABULAR TECHNIQUES

REFERENCE MANUAL

February 8, 1961
Ref. No. 1B3

Compiled by:
B. Grad
T. B. Glans

INTERNATIONAL BUSINESS MACHINES CORPORATION

White Plains, New York

TABULAR TECHNIQUES SEMINAR

AGENDA

Date: Wednesday, February 8, 1961
Time: 9:30 a.m. to 4:15 p.m.
Place: Roger Smith Hotel, East Post Road, White Plains, N. Y.

9:30 a.m.	History and Current Status	B. Grad
10:00	Coffee Break	
10:15	Limited Entry Tables	
	Sutherland & Company	T. B. Glans
	Hunt Foods and Industries	W. M. Selden
	Eastman Kodak	W. M. Selden
11:00	Work Problems and Discussion	T. H. Cleary
12:00 p.m.	Lunch	
1:00	Extended Entry Tables	
	GE TABSOL	B. Grad
	CODASYL	B. Grad
1:45	Work Problems and Discussion	T. H. Cleary
2:45	Coffee Break	
3:00	Review of Other Work	
	Northrop - North American - 9 PAC	B. Grad
3:15	Discussion of Plans and Future Programs	
4:15	Termination	

TABULAR TECHNIQUES SEMINAR

February 8, 1961

ROSTER

Lee Baker
Corporate Staff
Yorktown Heights, New York

Jane Bendall
World Trade Corporation
New York, New York

Kenneth R. Blake
Advanced Systems Development Division
White Plains, New York

No Richard L. Cline
Data Processing Division
New York, New York

Perry O. Crawford, Jr.
Advanced Systems Development Division
White Plains, New York

Roy Goldfinger
Corporate Staff
White Plains, New York

Julien Green
General Products Division
White Plains, New York

David Holstein
Corporate Staff
Yorktown Heights, New York

No William L. Kelly
Data Processing Division
Poughkeepsie, New York

Andrew Kinslow
Advanced Systems Development Division
White Plains, New York

William Korwan
Service Bureau Corporation
New York, New York

Millie Kohlman
~~Andrew H. Kruse~~
Data Processing Division
Syracuse, New York

Anthony A. Lea
Data Processing Division
Atlanta, Georgia

Lucille Lee
Data Processing Division
New York, New York

David Macklin
General Products Division
New York, New York

Sheila Mulroy
Data Processing Division
Rochester, New York

Harry Nagler
Data Systems Division
New York, New York

James A. Painter
Data Systems Division
Poughkeepsie, New York

Merwin D. Rayner
Data Processing Division
Beverly Hills, California

Stanley G. Reed
Data Systems Division
Poughkeepsie, New York

Samuel Reynolds
General Products Division
New York, New York

A. Owen Ridgway
Federal Systems Division
Bethesda, Maryland

ND James G. Robertson, Jr.
Advanced Systems Development Division
White Plains, New York

David Sayre
Corporate Staff
New York, New York

ND Walter M. Shenko
Data Systems Division
New York, New York

Roger Smith
Data Systems Division
New York, New York

Richard TenDyke
Advanced Systems Development Division
White Plains, New York

Mary Smith
Data Processing Division
Syracuse, N.Y.

ND Donald G. Thoroman
Data Processing Division
White Plains, New York

ND Frank A. Williams
Corporate Staff
White Plains, New York

Harold Sobcov
Data Systems Division
White Plains, New York

SPEAKERS

Thomas H. Cleary
Data Processing Division
Poughkeepsie, New York

Thomas B. Glans
Corporate Staff
Yorktown Heights, New York

Burton Grad
Corporate Staff
Yorktown Heights, New York

William M. Selden
Corporate Staff
Rochester, New York

R Blair Smith
Data Processing Division
White Plains, N.Y.

PURPOSE

The term "tabular form" as used here, is concerned with two-dimensional tabular layout, where position of information has significance in two directions for sequence control and display purposes. We are particularly concerned with the use of tabular form in programming for computers, and in describing decision and systems logic. Tabular form clearly associates conditions and actions through position. It may use virtually any existing language, from the most machine oriented to the most general. The difference is not in the use of naming or particular notational schemes, but rather the actual physical layout in which the program or system description is recorded.

The material presented in this Clearinghouse Report contains presentations of the work done by others, plus explanatory work problems to help characterize the approach. The material is organized to correspond with the agenda of the seminar. There has been no attempt made to edit, clarify or validate the particular information contained within each of the presentations.

The primary reason for holding a seminar on tabular techniques is to acquaint selected professional IBM personnel with the development of tabular display methods used for programming and systems description. In presenting the divergent techniques already in use we are not recommending any in particular, but only trying to convey understanding to those in IBM who need to evaluate this material. Then, they will be in a position to perform the necessary analysis and experimentation. IBM as a leader in the field must have people who are knowledgeable in this area so as to obtain an unbiased evaluation of the potential and opportunity of this approach.

A PROPOSED PROGRAM FOR RESEARCH ON TWO-DIMENSIONAL PROGRAMMING CONCEPTS

Burton Grad
Programming Systems
International Business Machines Corp.

March 1, 1960

TABLE OF CONTENTS

References

(1) D.D. McCracken, et al; Programming Business Computers; J. Wiley, 1959

(2) B. Grad and R.G. Canning; Information Process Analysis; Journal of Industrial Engineering; November-December, 1959

(3) Kemeny, Snell, Thompson; Introduction to Finite Mathematics; J. Wiley, 1958

(4) Harold Wolpe; An Algorithm for Analyzing Logical Statements to Produce a Truth Function Table; ACM Communications, March 1958

(5) Orren Y. Evans; Advanced Analysis Method for Integrated Electronic Data Processing (Draft); not published, 1959

(6) R. W. Murphy; A Definition of Block Diagrams; IBM Report IR-00065, 1956

(7) J. Jeenel; A Standardized Representation for Business Problems; Watson Research Laboratory Report, 1958

Section A - Two-Dimensional Programming

It is the purpose of this paper to discuss the concept of two-dimensional programming. This implies some non-serial programming structure to permit taking advantage of the ability of people to see relationships in two-dimensional form. While it is true that a sequence of statements can describe uniquely any operational procedure, this is really not the most important criterion. The two critical elements are:

(1) Does the representation technique provide for ease in preparation and communication? Is it a "natural" form for humans to prepare?

(2) Does the representation technique provide advantages in terms of preparing appropriate Processors? In other words, will the Processors be simple or faster or will the Processor running time be lessened or will the resultant object program be faster in operation or require less memory space.

Examination of tools used to date by systems designers, procedures analysts and computer programmers gives a revealing insight into the desired structure of a representational scheme which is properly "human-engineered". There are four popular forms currently in use for systems description:

(1) Schematic Flow Charts: These illustrate, in essentially a two-dimensional form, the significant systems elements, using lines and connectors to show interrelationships among these elements. The concept of precedence is established either through converting the lines to arrows or by conventions such as flow from left to right and top to bottom. This form has been extensively used by computer programmers and factory layout personnel. Often special symbols are adopted to represent a particular class of operation and typically extensive abbreviation is required to fit the procedural description within the available symbol forms. A sample schematic flow chart is shown on page B-5. A good explanation of this type of charting is given in chapter 3 of reference (1).

(2) Serial Flow Charts: This technique permits only one direction of flow, often from top to bottom. Again, various symbols are used to characterize the different types of operations and special coding is introduced to handle this reference to branch procedures. There are a number of minor variations on this basic theme, but all share the common concept of restrained arrangement of symbols. Serial flow charts are used for process descriptions and paperwork procedures diagrams. One such system is described in detail in reference (2).

(3) Logical Equations: Used primarily by design engineers for complex electronic equipment, the application of Boolean Algebra has grown considerably since 1940. Tending to be highly symbolic and abstract, this format permits various sophisticated techniques to be applied leading to systems minimization. Unfortunately, this approach (together with the extensive use of algebraic formulas) apparently leaves most non-technical personnel somewhat dubious and does not provide a suitable means for communication and reference. Conversely, the essential simplicity and analytic structure of logical equations do much to recommend it.

(4) Tabular Arrangements: In some areas, a tabular form has been adopted in order to clearly show the relationships between sets of conditions and sets of actions or results.

Since this paper is centered around the concepts of two-dimensional programming as embodied in tabular arrangements, we will explore a number of examples of this type of approach.

The foundation for much of the current work can be traced to the logical truth table as described in reference (3). Though used as an analysis tool (rather than directly for programming), this format has offered systems designers a technique for avoiding ambiguity and insuring comprehensiveness. Used in conjunction with logical equations, it provides a clear, easy-to-understand framework for describing and communicating analysis material. In general, a truth table consists of a series of columns in which the independent variables are used as column titles and the various combinations of Truth (T) and Falseness (F) of these variables are itemized in these "condition" columns (see Figure 1).

Figure 1

$(a \vee b) \wedge (a \overline{\wedge} b) \rightarrow c$

a	b	a∨b	a∧b	$a \overline{\wedge} b$	c
t	t	t	t	f	f
t	f	t	f	t	t
f	t	t	f	t	t
f	f	f	f	t	f

These columns are separated from a series of intermediate columns by a double line. These intermediate columns are titled by the particular portion of the initial equation whose truth or falseness is being analyzed. This is always done from the simplest to the most complex relation. Finally, the result is again separated by a double line and marked true or false as appropriate.

There is a sense of totality and straight forwardness in this format which is appealing to many systems analysts. For example, Harold Wolpe of IBM (reference 4) used a form a truth table to explain the operation of a relatively elaborate algorithm which he devised for automatically handling logical equations.

A direct outgrowth of this concept is described in reference (5) by Orren Evans of Hunt Foods and Industries. As part of his excellent paper describing a comprehensive set of techniques for systems analysis (systems flow charts, data layout, field definition, etc.) he uses a "Data Rule" concept. This is covered by an example shown in figure (2), below:

Figure 2

Rule No.	Prior Rule No.	Freq.	C_1	C_2	C_3	A_1	A_2	A_3
001		100	Y	Y		Y	Y	
002		30	Y	N	Y	Y		Y
003		5	Y	N	N	Y		
004		2	N				Y	

C_1, C_2 and C_3 each represent some conditional statement such as: Due~Balance + Amount~of~this~order ≤ Credit~Maximum. In each column a Y (for yes), and N (for no) or a blank(for "does not matter") is shown. To the right of a double line a series of Action columns are used. A_1, A_2, and A_3 each indicate some particular action like:

Mark Order "OK to ship"

A Y is used to indicate that this action is to be executed while a blank indicates that it is not to be carried out. Each row is called a Data Rule and has certain identifying material to the left of the condition columns. These are: rule number, which is the row number; prior rule number to indicate precedence relationships; and frequency, which denotes the number of times per week (or month or year) this particular data rule will be satisfied. The structure is such that one and only one rule can be satisfied for a given set of input values and the sequence of analyzing the Data Rules is not important in determining the proper Data Rule. This work has been presented to the Intermediate Range Task Force of CODASYL (Committee on Data Systems Language) and is presently being studied by this group.

While this form has much to recommend it from an analysis standpoint, there are a number of questions which can be raised concerning its usefulness as a programming device:

(1) Since each condition statement must result in either a "Yes" or a "No" answer, extra columns are needed to handle "Or" values and multiple ranges. For example, C_1 might represent: Marital-status is "single"; C_2: Marital- status is "divorced"; C_3: Marital- status is "married". Suppose the logic is as follows: If Marital - status is either "single" or "divorced", then put 1 in column 17; if Marital - status is "married", then put 2 in column 17. Figure (3) represents the Data Rule table needed for this problem.

C_1	C_2	C_3	A_1	A_2
Y			Y	
	Y		Y	
		Y		Y

It is apparent that this could become a serious problem as extensive multiple ranges entered the picture. It is also evident that slightly varying alternative actions can cause the same difficulty. This also may result in a more than linear increase in the number of rows required, since provision has to be made for all logical combinations of the conditions. On this basis, I believe that there is a major weakness in the handling of condition and action statements.

(2) The tables tend to be quite empty and extremely space consuming. In his write-up, Mr. Evans suggests one physical solution to this problem through multiple column identification. However, I don't believe this comes to grips with the underlying problem.

(3) The existence of a third state (blank for "not significant") prevents the direct use of a binary representation for the individual Data Rules. This binary coding would obviously offer very attractive memory reductions together with the possibility of direct binary word manipulation to detect the appropriate solution row. Further work in this direction might prove valuable.

(4) The table-to-table flow is not explicitly defined, therby leaving at least one critical aspect of a total data system open to question.

(5) Apparently, Commercial Translator Statements could be used as the language of the condition and action statements though this then requires the power of a Commercial Translator Processor to provide an object program.

(6) The connective between Condition Statements is only "AND" and the only sequence for executing action statements is that implied by the order of their listing.

In spite of these drawbacks, this technique does seem to offer many of the "human-engineering" advantages which we seek in a two-dimensional programming system:

(1) There is implicit indication of the path to be followed on successful or unsuccessful completion of a test. On success you continue across the current row. On failure you drop to the first test in the next row.

(2) There is a built-in error detecting function. If no solution is found, then failure on the last row could kick the program into a special error reporting routine.

(3) The truth table features aid in preventing and detecting logical errors or omissions.

(4) The formal structure is an aid to program communication.

(5) Through proper sequencing of the columns and Data Rules, a reasonably efficient operating procedure could be evolved.

There is other work in this direction which may be of use to us. For example, Bob Murphy of IBM proposed in 1956 a similar tabular technique for stating logical decisions without the restraint of explicitly defining all procedural sequences (as has to be done in a flow chart). His proposed technique had the same general properties as the Evans' work described above except that he used 0 for N and 1 for Y. He also experimented with a construction which permitted multiple success rows. The concepts which underly this work are described in reference (6). In a different direction, he explored briefly the use of a single column to represent multiple states or ranges of a particular variable. This is shown in figure 4. It appears that this might solve one of the serious problems in the Evans' approach.

Figure 4

Marital Status	A_1	A_2
Single	X	
Divorced	X	
Married		X

In 1958, Joe Jeenel, also of IBM, proposed a system delineation technique which included a modified truth table for logical decision rule description. He also presented a tabular approach to the control of program segments and loop hierarchies. This concept is explained in reference (7).

In 1957, Perry Crawford of IBM led an extensive study involving a full description of the various procedures involved in a particular customer application. In certain parts of the system, the rules were so complex that a tabular definition of the logic was used. One of the charts is shown in figure 5 (next page).

This is a far more compact representation of the problem than could have been obtained through the Evans' technique. However, it still has numerous weaknesses in terms of ease in preparation, ease in understanding, and efficiency in processing and operating.

Another area of tabular development has been in the field of product standardization. There is a well-known form used called a Collation Chart. This is nothing more than a listing of values for various critical specifications in the top rows of the sheet (see figure 6 below) and the names of the parts down the left-hand column. In the various intersections, the appropriate drawing number is entered. A dot is used as a horizontal ditto mark. Oftentimes the quantity, if variable, will be shown within the intersection. Otherwise, it appears adjacent to the part name.

Figure 6

Collation Chart for Electric Clock

Voltage	110	110	220	220	220	220
No.of Hrs.	12	12	12	24	24	24
Radium Dial	No	Yes	No	Yes	No	Yes
Glass	37B40	.	.	.	.	.
Case	37B50	.	.	.	.	.
Face	37B60	37B61	37B60	37B61	37B62	37B63
Hands	37B70	37B71	37B70	37B71	37B70	37B71
Gears	37B80	.	.	.	37B81	.
Motor	37B90	.	37B91	.	.	.

SHIPPING SCHEDULE DETERMINATION

CONDITIONS:

Stockage	Delivery	Availability	Further Conditions	Action	Shipping Schedule Date
S	DI or DN	QA		Ship at once	Today
		QN		Back order	DRO
	DD	Not applicable	OH-QRP $\geq$ QO	Defer order without reserving	DD
		Not applicable	OH-QRP $<$ QO and DD $\geq$ DRO	Defer order without reserving	DD
		Not applicable	OH-QRP $<$ QO and DD $<$ DRO	Defer order without reserving	DRO
NS	DI or DN	QA	QA $\geq$ 1/4 QO	Ship at once	Today
			QA $<$ 1/4 QO	Defer order and reserve	Today + SLT
		QN		Suspend order and order replenishment	Today + SLT
	DD	QA	QA $\geq$ QO	Defer order and reserve	DD
			DD $\geq$ Today + SLT and QA $<$ QO	Defer order and reserve	DD
			DD $<$ Today + SLT and QA $<$ QO	Defer order and reserve	Today + SLT
		QN	DD $\geq$ Today + SLT	Defer order	DD
			DD $<$ Today + SLT	Defer order	Today + SLT

Figure 5

A similar approach has been used to simplify and standardize shop routines and time standards.

All of these tabular techniques offer a natural mode for a two-dimensional programming language. It seems apparent, though, that the use of the intersection blocks for more than just a true-false indicator would extend the span of the table and might provide significant memory reductions. Since there is a variety of particular problems within the framework of a computer program, it may be desirable to analyze tabular formats for each of the key modes of operation: Input, Output, Formula Evaluation, Decision-making, Search, File Maintenance, and Supervisory (Executive).

Given this background of material, the balance of the paper will be concerned with particular aspects of the problem of creating a suitable two-dimensional programming language:

(1) Section B discusses various address modes. It is evident that if a "fixed" format is to be used (like a table) then a standardized address (or operand) system will probably be required. The conclusion of this section is that a two-address logic seems to be a reasonable solution to a two-dimensional programming system.

(2) Section C is brief analysis of the concept of controlled two-directional branching and its impact on the instructions needed in a two-address system.

(3) Section D is concerned with relative addressing and "contained" constants. These techniques make a programming language easily separable so as to permit a segmental approach to debugging.

(4) Section E describes a suggested minimum language embodying the principles described in the previous sections and then briefly indicates a few of the more important extensions and sophistications possible.

(5) Finally, Section F recommends a study program aimed at developing a useful two-dimensional programming language.

Section B - TWO ADDRESS LOGIC

In considering a two-dimenstional programming scheme, the number of operand addresses can be of significance. Most computers have been constructed with a one-address logic. Examples include all of the IBM 700 Series, Univac I and II and Burroughs 205. The IBM 650 is a special case where a second address is included in the instruction word just for instruction sequence control. Various of the newer computers have used a multiple operand address logic. The IBM 305, 1401, and 1620 as well as the NCR 304, Honeywell 800 and Univac 1103 all use two or more operand addresses - sometimes the number of addresses is variable.

It is the purpose of this Section to explore one, two and three-address logic for various classes of instructions and to try to show certain of the advantages and disadvantages of each mode of operation.

Any instruction system must explicitly state in each instruction the operation to be executed and must also state, either explicitly or implicitly, the memory location of the field (s) to be operated upon. In a one-address machine the basic instruction format is:

x...x	x...x
Instruction Code	Field Address

In operation, the computer's control element recognizes the instruction code and executes it using the information stored at the field address indicated. The computer then proceeds to the subsequent instruction location (which may not be in numerical order). There are many variations on this theme with index registers, partial word definitions, additional control data, etc., but essentially all single address machines have this basic pattern.

For three-address machines, the basic construction is:

x...x	x...x	x...x	x...x
Instruction Code	Field Address A	Field Address B	Field Address C

The general mode of operation is for the computer to carry out the operation indicated by the instruction code on the information stored at locations A and B and then either store the results at location C or switch control to location C. The same comments relative to variations is also applicable here with the added complexities of indexing multiple fields, defining partial word lengths for multiple fields, etc.

A two-address machine would have its instruction word composed as follows:

x...x	x...x	x...x
Instruction Code	Field Address A	Field Address B

Operation would vary considerably depending upon the nature of the instruction code. To illustrate, various classes of two-address instructions are noted below; one suggested operation and mode for representative members of each class is then described:

1. Move instructions -
   any instruction which moves information from one location to another. Examples include:

   READ Move information from designated input source (A) to destination location (B).

   WRITE Move information from source location (A) to designated output unit (B).

   ASSIGN Move information from source location (A) to destination location (B).

   These instructions could, of course, specify the movement of partial words or multiple words at once.

2. Relational Instructions -
   any instruction which compares two fields of information.

   Compare Greater Test to see if the contents of Field A are greater than the contents of Field B.
   Many other relational comparisons are possible, smaller, equal, not equal, greater or equal, and smaller or equal. Their operation mode would be identical to the Compare Greater instruction. Another possibility would be to have a generalized Compare instruction which would set a series of binary indicators like greater, smaller, etc.

3. Branch instructions -
   any instruction which changes the normal sequence of instruction execution.

   Branch Based on the contents of location A either switch control to location B or continue the normal operational sequence. Location A might designate some type of memory which has been preset by a previous test such as equals, not equals, overflow, etc.

4. Arithmetic instructions -
any instruction which performs an arithmetic function like add, subtract, multiply, divide, exponentiation, sine, cosine, etc.

Binary operations:

ADD Add the contents of location A to the contents of location B. Store the result in location B. The same approach would be followed for subtract, multiply, divide or exponentiation.

Unary operations:

SINE Determine the sine of the contents of location A. Store the result in location B.

Certain normally binary operations can be restated as unary operations if it is useful because of frequency of application. (B)
For example: (A) can be restated as:
Square Root (A) if (B) = 1/2

Other examples of unary operations are those which are performed for decimal (or binary) point location or for format modification (either input or output).

Shift Right (I) Shift the information in location A to the right by a predesignated number of positions (I). Store the contents in location B.

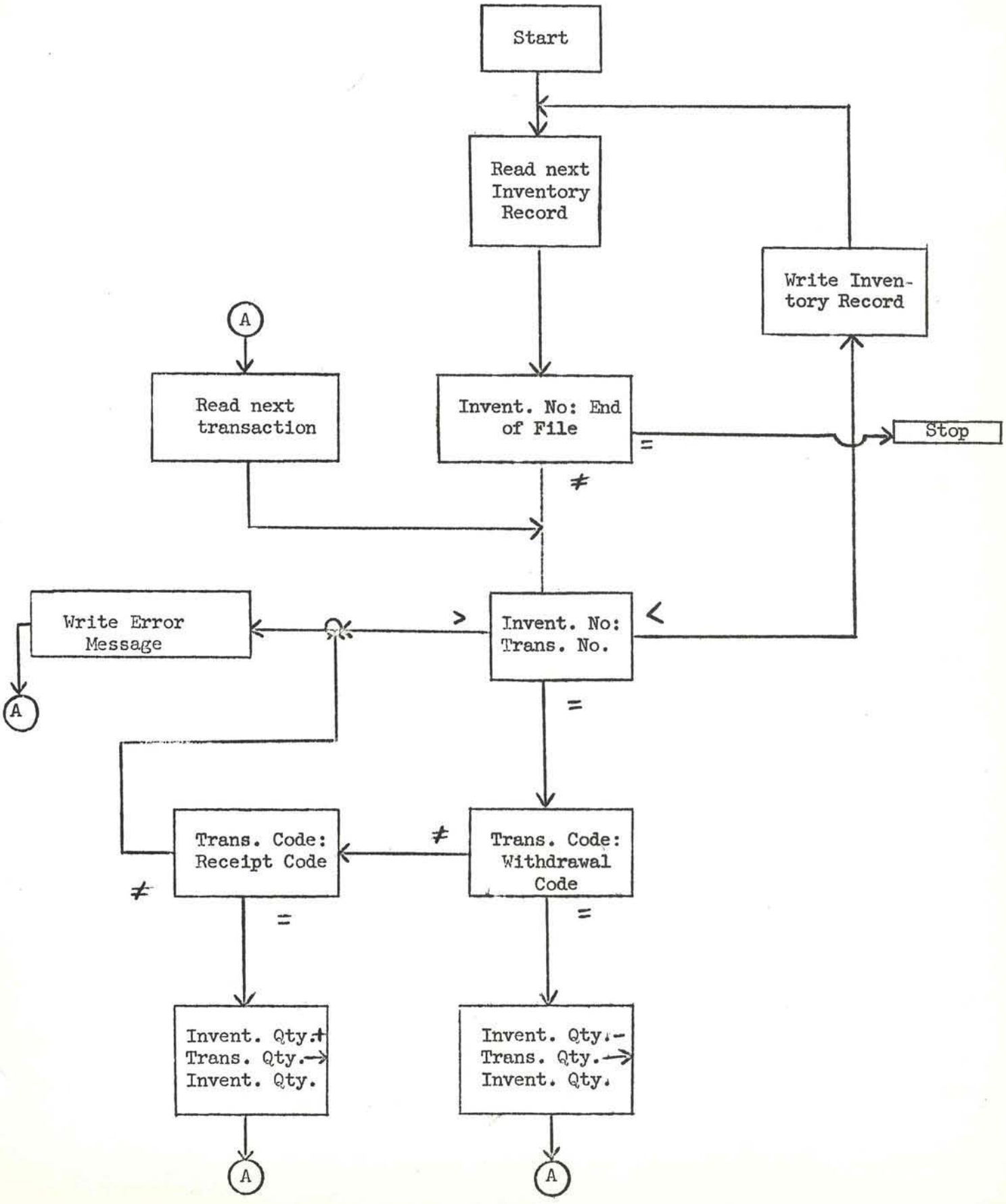
The same reasoning could be applied to shift left, shift right and round, shift left and test for overflow, etc. In arithmetic operations (and actually in any of the others) we can certainly consider the accumulator (s) as being merely special field locations so that there is no necessity for storing information in regular memory after an operation. For instance, MULT A, B where B was an accumulator address would simply mean to multiply the contents of the selected accumulator (B) by the contents of location A and store the result back in the selected accumulator (B).

5. Logical instructions -
any instruction which performs a normally defined Boolean function such as logical AND, logical INCLUSIVE OR, logical EXCLUSIVE OR, logical NOR, logical NOT, etc. The mode of operation would be similar to that of the arithmetic operations and could recognize both binary and unary logical instructions. Logical Not would be an example of a unary logical instruction. These could be used for control, masking, extracting, etc.

> Another possibility for logical operations might include the presetting of either a logically true or logically false indicator which could be tested in a subsequent Branch instruction. One implication of this type of operation is that the computer should be capable of operation in a binary number mode (0,1), though this is not necessary for the other operations. It suggests some way of defining structure at the bit level rather than at the character level. Definition could be implicit in the instruction code itself; for example, regular arithmetic might always refer to a 4 bits per character construction; but logical operations might always use a one bit per character construction; but on move and compare operations character construction would not be significant except as required for partial word operands.

This list is not an attempt to be definitive nor are the suggested modes of operation necessarily optimal for a given class of problems. Nevertheless, I believe that they show the comprehensiveness and potential scope of a two-address logic as well as indicating the simplicity and ease with which many frequent business data processing operations could be handled. It is also obvious that any programming system constructed with this logic could provide for any of the modifications possible in a one-address or three-address language, including: indexing, partial word selection, debugging stops, etc.

To examine further some of the potential advantages and disadvantages of this approach we might review the following example which has been coded in each address mode. I have assumed a simple mnemonic instruction code set for each configuration. Except for initializing, ending, and handling transactions with identification numbers greater than the largest valid inventory number, the problem is flow charted as follows:


Start
Read next Inventory Record
Write Inventory Record
A
Read next transaction
Invent. No: End of File
=
Stop
≠
Write Error Message
>
Invent. No: Trans. No.
<
A
=
Trans. Code: Receipt Code
≠
Trans. Code: Withdrawal Code
≠
=
=
Invent. Qty.+ Trans. Qty.→ Invent. Qty.
Invent. Qty.- Trans. Qty.→ Invent. Qty.
A
A

Within the framework of the flow chart and except for the start and stop routines I have delineated one possible program for solving this problem on a one-address machine:

Address	Inst. Code	Field Address	Comments
01	Read	Input 1	Move next inventory record to input buffer
02	Relocate	Invent. work area	Move info. in input buffer to invent. work area
03	Bring	Invent. No.	Move invent. no. into accumulator
04	Compare	End of File No.	Test accum. vs. end of file no.
05	Branch Equal	Stop Routine	If set proper comparison indicators equal, comparison indicator is "on", Stop Routine address
06	Bring	Invent. No.	_____
07	Compare	Trans. No.	Test accum. vs. trans. no., etc.
08	Branch Greater	28	To Transaction Error Routine
09	Branch Smaller	19	To write inventory record routine
10	Bring	Trans. Code	_____
11	Compare	Withdrawal Code	_____
12	Branch Not Equal	22	To Receipt Test Routine
13	Bring	Invent. Qty.	_____
14	Subtract	Trans. Qty.	Subtract trans. qty. from accum. result in accum.
15	Store	Invent. Qty.	Store contents of accum. at Invent. Qty. Location
16	Read	Input 2	_____
17	Relocate	Trans Work Area	_____
18	Branch Uncond.	06	_____
19	Relocate	Invent. Work Area	_____
20	Write	Output 1	_____

Address	Inst. Code	Field Address	Comments
21	Branch Uncond.	01	
22	Compare	Receipt Code	
23	Branch Not Equal	28	To transaction Error Routine
24	Bring	Invent. Qty.	
25	Add	Trans. Qty.	
26	Store	Invent. Qty.	
27	Branch Uncond.	06	
28	Relocate	Trans. Work Area	
29	Write	Output 2	
30	Branch Uncond.	06	

On the basis of 2 decimal digits for the instruction code and 4 decimal digits for the address, this program would require 30 x 6 = 180 memory location units.

In a similar way, I have prepared a possible program for a three-address machine.

Address	Inst. Code	Field Address A	Field Add. B	Field Add. C
01	Read	Input 1		Invent. Work Area
02	Compare Equal	Invent. No.	End of File No.	Stop Routine
03	Compare smaller	Invent. No.	Trans. No.	09
04	Compare greater	Invent. No.	Trans. No.	14
05	Compare not equal	Trans. Code	WithdrawalCode	11
06	Subtract	Invent. Qty.	Trans. Qty.	Invent. Qty.
07	Read	Input 2	_____	Trans. Work Area
08	Branch Uncond.	_____	_____	03
09	Write	Invent. Work Area	_____	Output 1
10	Branch Uncond.	_____	_____	01

(Cont.)

Address	Inst. Code	Field Add. A.	Field Add. B	Field Add. C
11	Compare Not equal	Trans. code	Receipt code	14
12	Add	Invent. Qty.	Trans. Qty.	Invent. Qty.
13	Branch Uncond.	_____	_____	03
14	Write	Trans. Work Area	_____	Output 2
15	Branch Uncond.	_____	_____	03

On the basis of two decimal digits for the instruction code and four decimal digits for each of the addresses this program would take 15 x 14 = 210 memory location units.

For comparison, the same problem is programmed for a two-address machine:

Address	Instruction Code	Field Address A	Field Address B
01	Read	Input 1	Invent. work area
02	Compare	Invent. No.	End of file no.
03	Branch	Equal	Stop Routine
04	Compare	Invent. No.	Trans. No.
05	Branch	Smaller	12
06	Branch	Greater	18
07	Compare	Trans. Code	Withdrawal code
08	Branch	Not Equal	14
09	Subtract	Trans. Qty.	Invent. Qty.
10	Read	Input 2	Trans. work area
11	Branch	Uncond.	04
12	Write	Invent. work area	Output 1
13	Branch	Uncond.	01
14	Compare	Trans. Code	Receipt code
15	Branch	Not equal	18
16	Add	Trans. Qty.	Invent. Qty.
17	Branch	Uncond.	04

(cont.)

Address	Instruction Code	Field Address A	Field Address B
18	Write	Trans. work area	Output 2
19	Branch	Uncond.	04

On the basis of two decimal digits for the instruction code and four for each address, this program would require 19 x 10 = 190 memory location units.

Let's examine these programs at least superficially to draw some tentative conclusions.

1. There is no striking difference in memory space required for either of the three programming modes.

2. The three-address mode does well because of the arithmetic capability and the combined compare and branch instruction.

3. The one-address system is a little more difficult and time consuming to write and requires more words of instructions (though not necessarily more memory space, dependent on the internal word structure).

4. Two-address logic does very well on move type instructions (read, write) and on "add-memory" operations.

5. There were 13 different instruction codes used for the one-address program. The three-address system used nine different instruction codes while the two-address system used only six. This is not necessarily significant, but it may be indicative of a somewhat simpler instruction code structure.

6. Almost half (9) of the 19 instructions in the two-address system were branch instructions. Suppose it were possible to change the concept of the compare instruction so that a specific indicator was examined to see if it was on or off, and suppose that the "success" branch was always the next regular instruction while the "failure" branch was a fixed interval away; then it would be feasible to eliminate virtually every branch except where three or more alternate exits existed or where the branch was unconditional. In the program under

discussion, this would have eliminated four branch instructions. This may be feasible to accomplish through the two-dimensional programming approach. This aspect of success-failure physical location will be discussed in the next section.

On the basis of these comparisons, I believe it is evident that intensive study of two-address programming systems may offer important ways to reduce computer logic cost while providing more efficient programming instructions.

Section C- Controlled Two-Directional Branching

In normal programming methods with a one-address or multiple-address machine, the succeeding instruction in serial sequence is always implied as the alternate address on a branch instruction. The explicit branch is stated directly in the instruction. This is all that can be expected of any one-dimensional programming scheme.

In contrast, a two-dimensional programming system implies a two-dimensional branch. If the test succeeds, then the proper subsequent instruction follows next in the same row (or column). If the test fails, then the subsequent instruction is the first test in the next row (or column). Since the total number of columns per row is known, it is a straight-forward matter to compute the next instruction location for a test failure.

With this concept, we can think of a two-address instruction of "compare greater", which implicitly defines the success instruction address and the failure instruction address. This would require defining a complete set of the Compare instructions, which were of significance (greater, smaller, equal, not equal, greater or equal, smaller or equal).

Avoided would be the definition of any Branch instructions. Using this approach, the two-address program for the simple inventory problem used as an illustration could be reduced by eliminating 03, 05, 08, 15. However, it would require Compare Equal, Compare Smaller, Compare Greater and Compare Not Equal instructions. With this change, the program would be reduced to 15 instructions; with 10 characters each, 150 memory location units would be needed.

Section D - Relative Addressing and "Contained" Constants

All digital computers which have been announced up to now have had an instruction structure, which has called for stating one or more specific operand addresses. Some provision has been made for modification of these through the use of index registers, but the operating program presupposes absolute addresses.

In preparing new programming languages (symbolic assembly programs, compilers, etc.), one of the major efforts has been to enable the programmer to avoid this fixed address assignment. Two basic approaches have been taken to solving this problem. The first and less sophisticated, is to use a relative address like V014, which means the 14th word after that location designated as V000. This enables the program to be segmented, yet during the compiling stage, there is just the quite simple job of calculating the actual address of V014 as the location of V000 + 14. This is a very common practice. It does require, however, that the programmer in effect sub-structure the memory assignment and remember to use the correct relative address whenever he refers to that information.

A second approach has attempted to improve this area further. This is the concept adopted for FORTRAN and the Commercial Translator Language. Here, a mnemonic code name is assigned to the information field. For Example, EMPNO might refer to the Employer Pay Number field. FORTRAN restricts this to a six (6) character code. Commerical Translator allows the use of up to thirty characters plus adjectival modification to indicate file and record hierarchy. Because of the mnemonic aid,it is expected that the programmer will have far less trouble writing the correct pseudo-address, which the field name, of course, has become. During the compiling, each of these names will be assigned an actual address and each time the name occurs, this same actual address will be assigned. This is significantly easier for the programmer, particularly in explaining or communicating the program to someone else. It would also be a great aid in debugging except that the program debugs at the machine language level, which implies fixed addressing; this in turn means that the programmer has to convert from the absolute address to the field name that he has been using.

In preparing machine language programs, it has also been historically necessary to store any constants required and then call them out through using the appropriate absolute address. With relative addressing,the problem is only helped slightly since a memory location must still be used to store the needed constant. With FORTRAN, etc., the programmer may use a constant directly in his instructions; e.g. Y = 16 X. The compiler assigns this constant a location and, in the object program, refers to this location. While this is a big improvement for the programmer, it still uses up memory space for all the various constants needed. One interesting variation is to use the addresses themselves as constants; this yields the most commonly used integers. It is proposed in this paper that a new programming language be constructed so as to permit direct use of mnemonic addressing and so as to contain constants within the instruction word, thereby requiring no additional memory space for their storage.

(cont.)

It is further proposed that during compiling the computer may assign a relative address to each mnemonic address and that the internal computer logic should be structured so as to operate directly on this relative address. Furthermore, that the contained constants, where possible, be retained within the instruction and not independently stored. This would require that the instruction code recognize that one of the operands was a constant and not a mnemonic address.

If we consider these suggestions in context with the two-address logic recommendation, we can consider that the instruction structure for the programmer might appear as follows:

1. Move Instructions

Move (Field Name A) to (Field Name B)

or

Move Constant to (Field Name B)

2. Relational Instructions

Compare (Field Name A) with (Field Name B)

or

Compare Constant with (Field Name B)

3. Branch Instructions
Not affected

4. Arithmetic Instructions

Add (Field Name A) to (Field Name B)
[and store in (Field Name B)]

or

Add Constant to (Field Name B)
[and store in (Field Name B)]

Unary arithmetic would also have both configurations.

5. Logical Instructions
Similar to the arithmetic operations.

During compiling, the only changes would be to convert the operation name to an operation code and translate the field names into relative addresses. For example:

	Move	(Field Name A)	to	(Field Name B)
might become	27	V001		V009
	Move	4728	to	(Field Name B)
might become	28	4728		V009

The major reason for converting to a relative address is that current-day machines lose significant time in performing a dictionary look-up operation. It is quite conceivable, of course, that new machine components may change this picture considerably in which case it might be advantageous to store the mnemonic address rather than a relative location.

A major concern, which has been alluded to earlier is the difficulty of debugging in a machine oriented language. It is therefore believed worth considering the preparation of interpretive programs, which allow the computer to debug in the systems oriented language itself. Once the program has run satisfactorily, then the regular compiler could be used to prepare a set of instructions suitable to the particular computer. To generalize this point it would, for example, seem worthwhile to construct an interpretive FORTRAN processor which would permit direct debugging in the FORTRAN language, even though eventually an object program will be compiled. Such programs should be relatively inexpensive to prepare and will increase significantly the desire of many experienced programmers to use these advanced programming languages. This concept is tied somewhat to the idea of relative addressing and contained constants, since with these two tools the interpretive program can be quite simple and the analysis of results quite straight-forward.

Section E - A Suggested Minimum Two-Dimensional Language

For the purpose of establishing a common frame of reference, a "minimum" language is described, which would enable a computer to rapidly execute the bulk of the operations, which seem to be required in two-dimensional programming. There is obviously nothing magical about this particular set of instructions. However, industrial computing experience indicates that these would permit a one to one translation of many relations and actions into computer instructions. The original set has intentionally omitted indexing, partial or multiple word operations, and logical (Boolean) manipulations. These items are discussed later in this section. In the symbolic statements included after each instruction definition, () means "contents of location designated" and → means "replaces".

Move Instructions

READ A, B

Move the next record from input source A to a series of internal locations beginning with location B. Source A may be a card reader, punched tape reader or any selected magnetic tape transport. This instruction could operate with fixed record length (N words or characters or bits), variable record length (separated by a recognizable record mark), or with defined record length as a modifier to the READ instruction (such as Read 60 to imply Read 60 consecutive words).

(A) → (B)

WRITE A, B

Move the next record from a series of internal locations beginning with location A to output destination B. Destination B may be a card punch, tape punch, printer or any selected magnetic tape transport. The same statements about record length that were made in the READ instruction would apply to the WRITE instruction.

(A) → (B)

ASSIGN A, B

Move the contents of the Field A designated location to the Field B designated location; this instruction moves information from one memory location to another memory location.

(A) → (B)

ASSIGN CONSTANT A, B

Move the contents of Field A to the Field B designated location; this instruction transfers a constant to a memory location.

A → (B)

(cont.)

COMPARE EQUAL A, B

Compare the contents of the Field B designated location with the contents of the field A designated location. If these are identically equal, then branch to the preassigned success location (S); if these are not identically equal, then branch to the preassigned failure location (F).

if (B) = (A) then next instruction at S

if (B) = (A) then next instruction at F

COMPARE SMALLER A, B

if (B) < (A) then go to S

if (B) $\geq$ (A) then go to F

COMPARE GREATER A, B

if (B) > (A) then go to S

if (B) $\leq$ (A) then go to F

COMPARE NOT EQUAL A, B

if (B) $\neq$ (A) then go to S

if (B) = (A) then go to F

If the machine's internal characteristics lend themselves to a slightly more elaborate mode of operation, then the instruction set may also include:

COMPARE SMALLER OR EQUAL

COMPARE GREATER OR EQUAL

COMPARE CONSTANT EQUAL A, B

Compare the contents of the Field B designated location with the contents of Field A. If these are identically equal, then branch to the preassigned success location (S); if these are not identically equal then branch to the preassigned failure location (F).

if (B) = A then go to S

if (B) $\neq$ A then go to F

COMPARE CONSTANT SMALLER A, B

if (B) < A then go to S

if (B) $\geq$ A then go to F

COMPARE CONSTANT GREATER A, B

if (B) > A then go to S

if (B) $\leqslant$ A then go to F

COMPARE CONSTANT NOT EQUAL A, B

if (B) $\neq$ A then go to S

if (B) = A then go to F

If the internal machine characteristics lend themselves to this mode of operation, the instruction set may also include:

COMPARE CONSTANT SMALLER or EQUAL

COMPARE CONSTANT GREATER or EQUAL

GO TO A

transfer program control to location designated in Field A. This is an unconditional branch instruction. The reason it is needed rests with the limitations of a two-dimensional programming system (in contrast to an n - dimensional system).

Arithmetic Instructions

ADD A,B

Add the contents of the Field B designated location to the contents of the Field A designated location. Store the result in the Field B designated location. Either Field A or Field B may be a special high-speed accumulator.

(B) + (A) $\rightarrow$ (B)

SUBTRACT A, B

(B) - (A) $\rightarrow$ (B)

MULTIPLY A, B

(B) x (A) $\rightarrow$ (B)

DIVIDE A, B

(B) $\div$ (A) $\rightarrow$ (B)

We have omitted in this instruction set exponentiation or unary arithmetic operations. In the problems which we have handled to date, their frequency of use has been such that this need could be best served through use of stored subroutines.

ADD CONSTANT A, B

Add the contents of the Field B designated location to the contents of Field A. Store the result in the Field B designated location.

(B) + A → (B)

SUBTRACT CONSTANT A, B

(B) - A → (B)

MULTIPLY CONSTANT A, B

(B) x A → (B)

DIVIDE CONSTANT A, B

(B) ÷ A → (B)

SHIFT LEFT A, B

Shift the contents of the Field B designated location to the left by A positions. Conceptually this operates on a predefined word length. The result is stored in the Field B designated location. If there is an overflow consider this as a failure; if no overflow, this is a success.

SHIFT RIGHT A, B

Shift the contents of the Field B designated location to the right by A positions. The result is stored in the Field B designated location.

If desired, a SHIFT RIGHT and ROUND instruction may be included.

Logical (Boolean) Instructions have been omitted.

Miscellaneous Instructions

NO OPERATION

STOP

Simple Extensions of the Minimum Language

Among additional features which may be desirable in a two-dimensional programming system are: address indexing; partial word movement; multiple word movement; error control (debugging stops and input variance detection); extended arithmetic capability including square root, integral exponentiation (square and cube) and certain trigonometric functions (sin, tan). A simple approach to some of these features is described in this section.

One problem in address indexing has been the attempt to consider multiple subscripts. Writing a compiler to automatically recognize and handle multiple subscripts is a relatively complex chore. A simple concept which handles any number of subscripts is through a calculated index which is the function of the subscript values.

INDEX ASSIGN I,A,B

Modify the Field A designated location by the contents of the Field I designated location. Move the contents of this modified Field A designated location to the Field B designated location.

(A (I)) (B)

ASSIGN INDEX I,A,B

Modify the Field B designated location by the contents of the Field I designated location. Move the contents of the Field A designated location to the modified Field B designated location.

(A) (B (I))

The contents of the Field I designated location can, of course, be previously defined by any acceptable operation. Let's assume we have a three dimensional matrix stored in consecutive locations first by column, then by row, then by table. There are three parameters to be defined: R (max Rows), C (max Columns), T (max Tables). We will define r as the row subscript, c as the column subscript, and t as the table subscript.

I f (r,c,t)

I Initial Location r cR tRC

Test for completion would be I versus Initial Location - R (1-C(1-t)). This principle can, of course, be extended to any n-dimensional subscripting plan. It could, of course, also be applied in concept to any arithmetic or comparison operations.

Partial word movement could be handled through a similar approach:

PARTIAL ASSIGN I,F,A,B

Move the Ith (Initial) through the Fth (Final) position

of the contents of the Field A designated location into the Field B designated location starting at the left-most position.

ASSIGN PARTIAL I, F, A, B

Move the contents of the Field A designated location starting from the left-most position into the Ith through the Fth position of the Field B designated location.

Multiple word movement could be handled as follows:

MULTIPLE ASSIGN M, A, B

Move M words starting at the Field A designated location into a series M words beginning at the field B designated location.

Error Control for debugging stops and input variance detection could be handled as follows:

ERROR GO TO A, B

Transfer control to the Field A designated location. Set up to return control to the Field B designated location upon completion of processing the next table.

Unary Arithmetic could follow the following form:

SQRT A, B

Find the square root of the contents of the Field A designated location; store the result in the Field B designated location.

Recommendations

It seems evident that present approaches to systems-oriented languages do not appear to be capable of making a basic breakthrough in the one really critical programming problem: Systems description. Until a technique is developed which supersedes flow charting and yet is readily computer-understandable we shall not have achieved an effective application language.

Because the logic of two-dimensional programming seems irrefutable from a users standpoint it certainly seems worthwhile to aggressively pursue a research and development program aimed at exploring and advising techniques in this general area. The program below would, I believe, have a reasonably good chance at full success:

(1) Propose a specific two-dimensional language based on the Hunt Foods', Aeronutronics and IBM work (particularly the direction suggested by Perry Crawford). The language may differ for various types of usage (Input, Output, File Maintenance, Decision-making, Arithmetic, etc.)

(2) Prepare a simple interpretive program to solve two-dimensional programs. Also prepare a converter for translating from a Commercial Translator level of word choice to a machine-oriented level.

(3) Program a variety of problems using the language. Try to teach the language to non-programming personnel; have them write programs for areas of knowledge using the language.

(4) Run a few trial problems with the programs written in the two-dimensional language. Explore techniques for debugging and program modification.

(5) Revise the language; this may include permitting, but not requiring, certain desirable special features like SORT, UPDATE, MERGE.

(6) Prepare a "bootstrapped" compiler for two or more different computers. This will give a deeper insight into the language structure. Prepare a new Interpreter and Converter.

(7) Conduct further experimentation bringing in appropriate Field Sales personnel.

(8) Prepare manuals on the language covering the following areas:

(1) primer
(2) reference
(3) application experience
(4) Interpretive, Converter, Compiler Programs.

(9) Publish the results and present to CODASYL or other appropriate professional groups.

This is obviously an ambitious program and would probably involve 3-5 full time people together with appropriate help from many others on a part-time basis. Because the need is so great, I feel that the time schedule should be intentionally brief with completion targeted at 12-18 months from initiation.

I would estimate that the total cost of such a project including computer time, programming, technical writing, outside consultants, office support, and salaries for full-time personnel (but not for part-time) would approximate $300,000.

If the project worked out as I would hope then the reward should be a major advance in the programming (in contrast to the coding) art. We would have a basic new tool for systems design and a firm basis for language standardization.

INFORMATION PROCESSING SYSTEM ANALYSIS

SUTHERLAND and CO.

INFORMATION PROCESSING SYSTEM ANALYSIS INSTRUCTIONS

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

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

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

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

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

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

(3) Description Sheets
   a. Input Description
   b. Process Description and Process Description Continuation
   c. Output Description

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

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

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

3. Input Description Sheet.

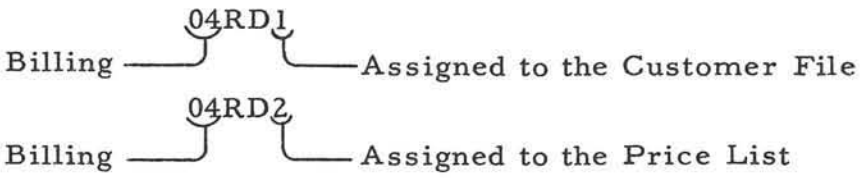
A. General. An Input Description Sheet is used to describe the content of Action Sets and Retained Data Sets which are input to the information system.

B. Headings.

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

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

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



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

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

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

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

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

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

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

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

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

D. Element Name.

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

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

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

F. Element Code - Suffix.

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

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

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

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

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

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

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

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

N. Identifying Element Codes.

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

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

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

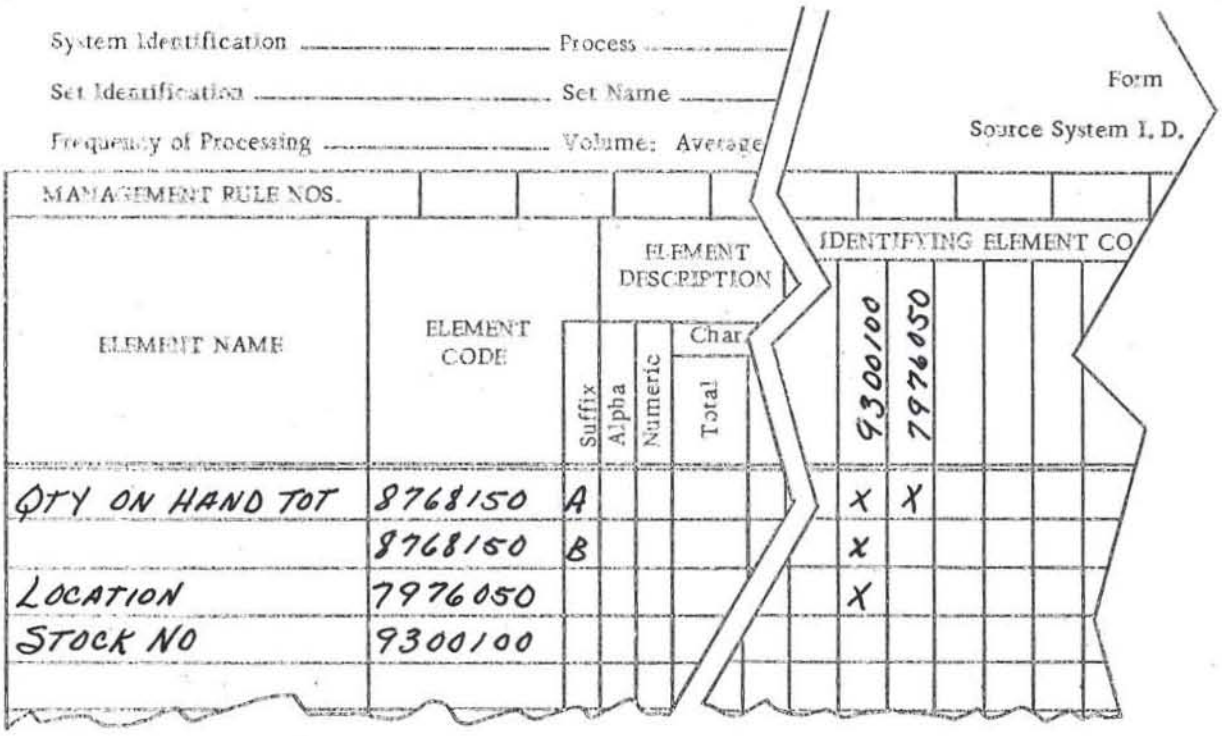


FIGURE 1. USE OF ELEMENT SUFFIXES AND IDENTIFYING ELEMENT CODES

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

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

4. Process Description Sheet.

A. General.

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

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

B. Headings.

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

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

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

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

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

Examples of line numbers are :

011
012
0131
0132
014

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

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

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

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

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

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

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

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

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

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

I. Source - Set Identification.

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

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

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

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

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

K. Operation.

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

symbols in the column:

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

(2) Explanation of Operation Symbols.

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

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

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

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

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

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

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

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

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

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

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

P. Operation.

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

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

(2) Explanation of Operation Symbols.

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

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

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

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

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

R. Management Rule Suffix and Frequency.

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

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

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

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

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

PROCESS DESCRIPTION

Page _____ Of _____

System Identification _____

Process _____ ☐ Validation

Prepared by _____

Date _____

LINE NO.	C/A	MANAGEMENT RULE NO. Current	MANAGEMENT RULE NO. Prior	SOURCE ELEMENT Name, Prior Result or Actual Value	SOURCE ELEMENT SUF.	SOURCE Set Ident. or Rule No. for Prior Result	COND. Greater Than	COND. Less Than	COND. Equal To	Oper (+ - / X Σ)	Set = To (V)	SOURCE/DISPOSITION ELEMENT Name, Result, Prior Result or Actual Value	SOURCE/DISPOSITION ELEMENT SUF.	SOURCE/DISPOSITION Set Ident. or Rule No. for Prior Result	Oper (+ - / X)	Note Ref. (V)	MANAGEMENT RULE SUFFIX AND FREQUENCY A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R
001	6	001		QUAN ON HAND		67491	✓		✓			QUAN ORDERED		6089			Y	N	Y	Y	N													
002				CREDIT RATING		40A			✓			(A)					Y	Y	N	N	N													
003				QUAN ORDERED		6089		✓	✓			CUSTOMER MAX		40A					Y	N													
																	8	0	1	0	0												
																	0	6	0	4	1												
004	A			QUAN ORDERED		6089					✓	QUAN SHIPPED		409257			X		X														
005				QUAN ON HAND		67491					✓							X															
006				(0)							✓									X	X												

FIGURE 2. EXAMPLE OF USE OF MANAGEMENT RULE SUFFIX AND FREQUENCY

Cond: Y = Condition Is Satisfied; N = Condition Is Not Satisfied; Blank = Condition Not Applicable

Action: X = Action to Be Taken; Blank = Action Not to Be Taken

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

(6) For a Management Rule, on the line (s) following the last line describing the conditions, the analyst will indicate the probable Frequency of Occurrence as a percentage for the results of the conditions listed under each suffix. The total Frequencies of Occurrence for all suffixes within a Rule should be 100 percent. For any frequencies less than 1%, use "1". In Figure 2 the Frequency of Occurrence is indicated between lines 003 and 004. In this example the probability of the conditions of rule 001A prevailing is 80%, written "$\overset{8}{0}$". For rule 001D, the probability of occurrence is 4%, written "$\overset{0}{4}$".

5. Process Description Continuation Sheet.

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

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

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

D. Management Rule Suffix and Frequency.

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

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

6. Output Description Sheet.

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

B. Headings.

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

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

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

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

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

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

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

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

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

C. Element Name.

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

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

D. Element Code.

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

E. Element Code - Suffix.

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

F. Element Description - Alpha.

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

G. Element Description - Numeric.

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

H. Characters - Total.

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

I. Characters - Decimal.

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

J. Element Classification.

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

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

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

L. Source - Set Type.

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

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

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

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

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

M. Source - Source Set Identification for Direct Recording.

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

N. Identifying Element Codes.

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

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

O. Reference Note.

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

APPENDIX I - DEFINITIONS

1. Action Element

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

2. Action Modifier Element

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

3. Action Set

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

4. Constant Value

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

A. Descriptive Constant

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

Examples are :

/ Blank / - Designates one or more blanks.

/ Current Year / - Designates 1962, if that is the current year.

/ ANNN / - Designates a field in which the first character is non-numeric and the rest are numeric.

B. Literal Constant

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

An example is :

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

5. Direct Recording

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

6. Frequency of Occurrence

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

7. Identification Element

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

8. Information Element

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

9. Management Rule

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

10. Output Set

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

11. Process

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

12. Recognition Element

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

13. Retained Data Set

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

14. Set

A meaningful grouping of more than one element of data.

15. Superfluous Element

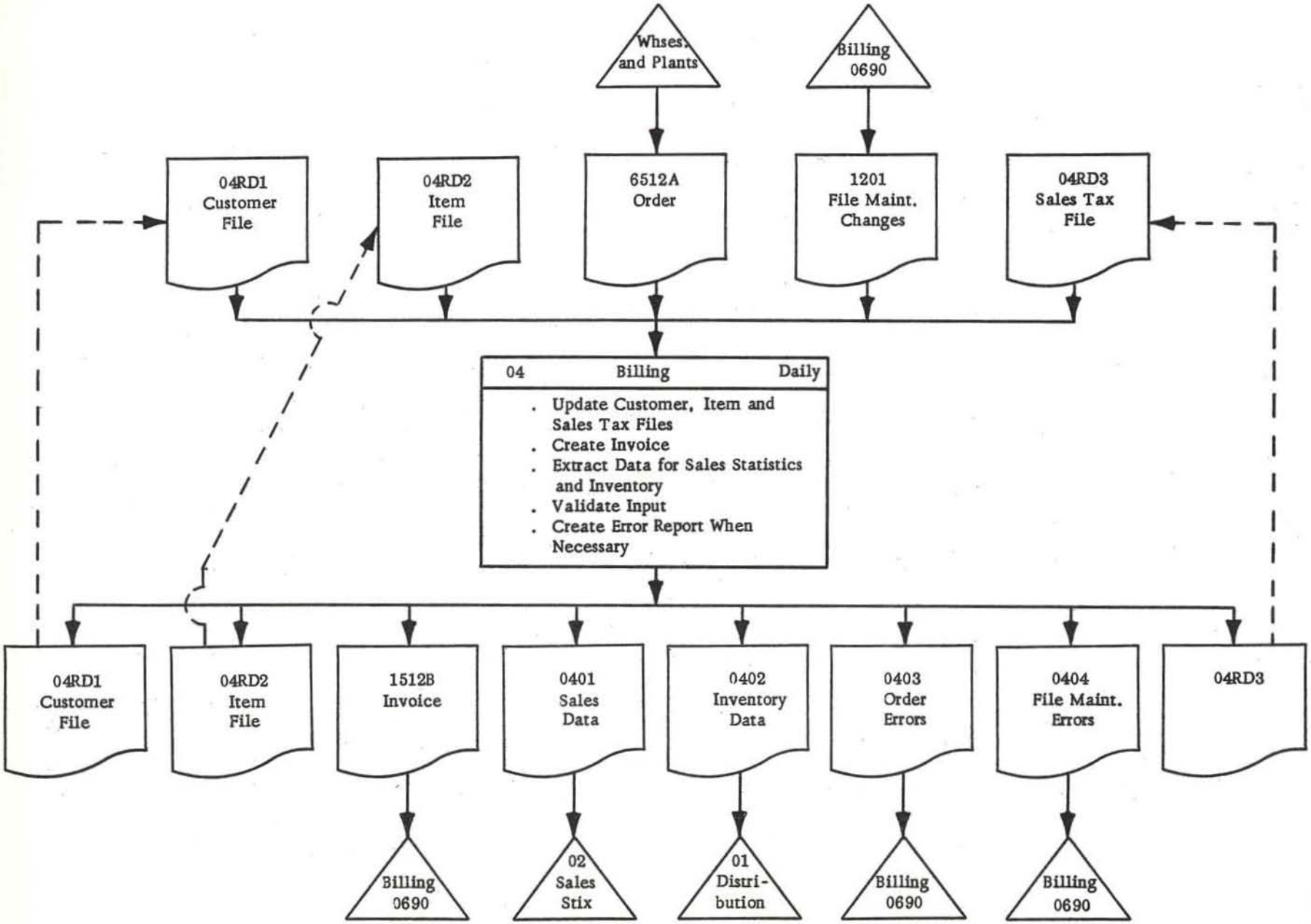
An element within an Action Set which is not required for processing in this or subsequent systems.

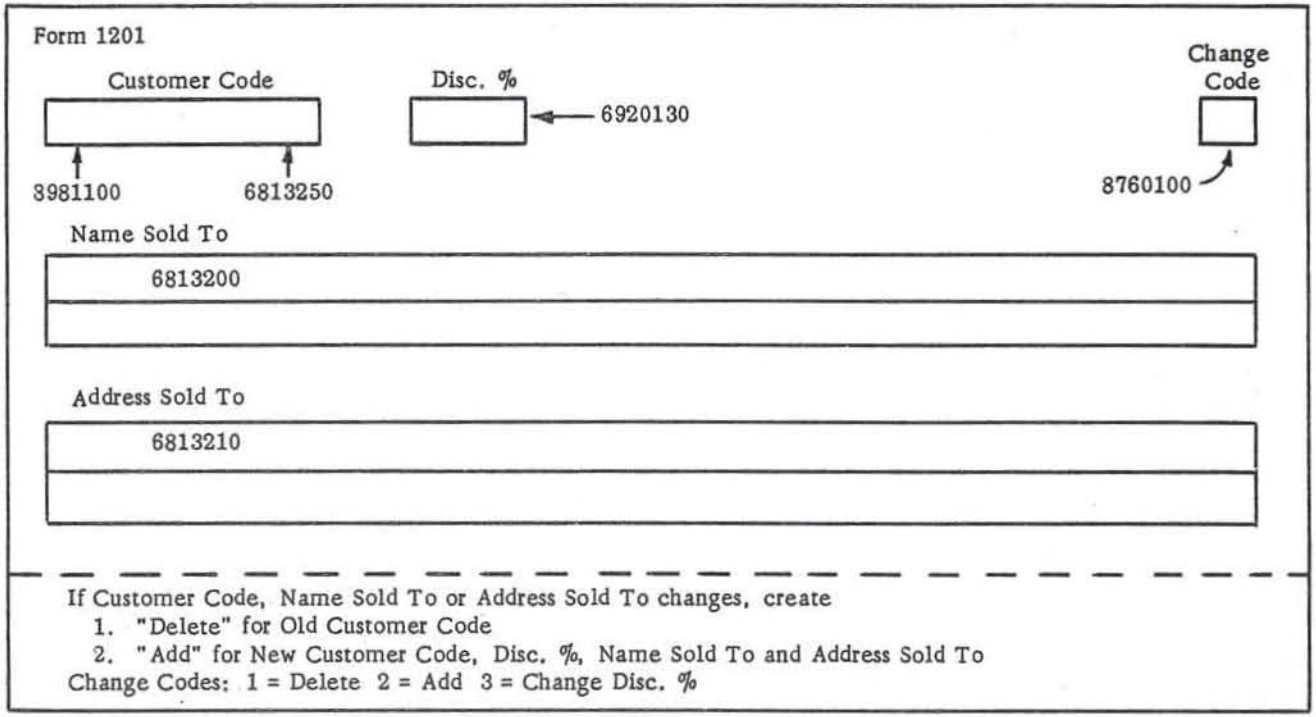
SYSTEM CODE 04 BILLING

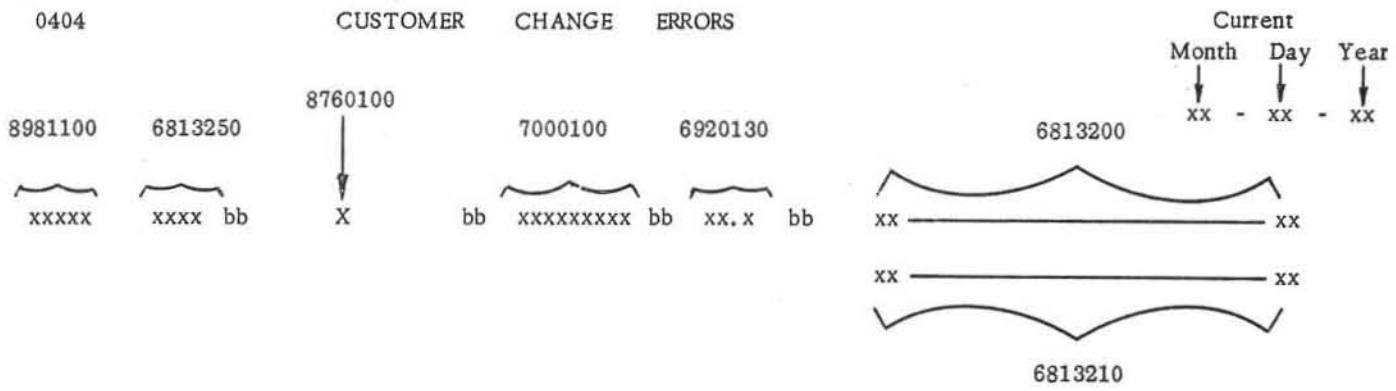
1. Purpose: To develop an invoice from a copy of the order which indicates that shipment has been made to a customer from a warehouse or factory.

2. Scope: The system will include all debit billing to all customers.

3. Other Outputs:
   a. Selected data will be furnished to the Sales Statistics system for Sales Accounting and Sales History.
   b. Selected data relative to inventory will be furnished to the Distribution system for inventory adjustments.
   c. A record of input received that did not meet the criteria established (invalid) will be furnished to the Billing Department.


Whses. and Plants
Billing 0690
04RD1 Customer File
04RD2 Item File
6512A Order
1201 File Maint. Changes
04RD3 Sales Tax File
04 Billing Daily
. Update Customer, Item and Sales Tax Files
. Create Invoice
. Extract Data for Sales Statistics and Inventory
. Validate Input
. Create Error Report When Necessary
04RD1 Customer File
04RD2 Item File
1512B Invoice
0401 Sales Data
0402 Inventory Data
0403 Order Errors
0404 File Maint. Errors
04RD3
Billing 0690
02 Sales Stix
01 Distri-bution
Billing 0690
Billing 0690


Form 1201
Customer Code
Disc. %
6920130
Change Code
3981100
6813250
8760100
Name Sold To
6813200
Address Sold To
6813210
If Customer Code, Name Sold To or Address Sold To changes, create
1. "Delete" for Old Customer Code
2. "Add" for New Customer Code, Disc. %, Name Sold To and Address Sold To
Change Codes: 1 = Delete 2 = Add 3 = Change Disc. %



Page Number xxxxx

System Identification OA Process BILLING **INPUT DESCRIPTION** Page 1 of 23

Set Identification 1512A Set Name ORDER Form Type MANUAL Prepared by: HCD

Frequency of Processing DAILY Volume: Average 500 Peak 1000 Source System I. D. 01 Date 6/15/60

MANAGEMENT RULE NOS. 006 007 008 009 010 011 012

ELEMENT NAME	ELEMENT CODE	ELEMENT DESCRIPTION: Suffix	Alpha	Numeric	Char. Total	Char. Decimal	ELEMENT CLASS. (X): Recog.	Ident.	Action	Action Mod.	Info.	Superfluous	NO. TIMES AN ENTRY MAY APPEAR ON THIS SET: Average	Peak	VALIDATION RULE (S)	IDENTIFYING ELEMENT CODES (X): 8368100	9900100	Ref. Note (#)	REMARKS
ACCT CENTER	6017100		A		13						X		1	1		X			
MO CUST ORD	8168620			N	02	0					X		1	1		X			
DA CUST ORD	6836620			N	02	0					X		1	1		X			
YR CUST ORD	9980620			N	02	0					X		1	1		X			
ORDER NO	8368100		A	N	09			X					1	1					
MO SHIPPED	8168650			N	02	0					X		1	1	001	X			
DA SHIPPED	6836550			N	02	0					X		1	1	001	X			
YR SHIPPED	9980650			N	02	0					X		1	1	001	X			
INVOICE NO	7680100		A	N	09						X		1	1		X			
CUST NAME SOLD	6813200		A		70							X	1	1		X			
CUST ADDR SOLD	6813210		A	N	70							X	1	1		X			
CUST NAME SHIP	6813220		A		70						X		1	1		X			
CUST ADDR SHIP	6813230		A	N	70						X		1	1		X			
CUST ORDER NO	6813100		A	N	12						X		1	1		X			
SALESMAN NO	8981100			N	05	0		X					1	1	003	X			
CUST ACCT NO	6813250			N	04	0		X					1	1	003	X			
TERMS PAYMENT	9496100		A	N	23						X		1	1		X			
SHIPG TERMS	9111100		A		23						X		1	1		X			
CARRIER NAME	6466100		A		24						X		1	1		X			
SHIP FROM TOWN	9110120		A		13						X		1	1		X			
SALES TAX CODE	9464160		A	N	05			X					1	1	002	X			
PRICE BASE	8664100		A	N	08						X		1	1		X			
SIZE	9144100		A	N	21							X	3	15		X	X		
LINE NAME	7941100		A	N	25							X	3	15		X	X		

System Identification 04 Process BILLING

INPUT DESCRIPTION

Set Identification 1512 A Set Name ORDER Form Type MANUAL

Frequency of Processing DAILY Volume: Average 500 Peak 1000 Source System I. D. 01

Prepared by: HCJ

Date 6/15/60

MANAGEMENT RULE NOS. 006 007 008 009 010 011 012

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

Appendix II, Page 6

System Identification 04 Process BILLING

INPUT DESCRIPTION

Page 3 of 23

Set Identification 04RD1 Set Name CUSTOMER FILE Form Type MAG T. Prepared by: HCU

Frequency of Processing DAILY Volume: Average 20,000 Peak 30,000 Source System I. D. 04 Date 6/15/60

MANAGEMENT RULE NOS.

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

Appendix II, Page 7

System Identification 04 Process BILLING

INPUT DESCRIPTION

Page 4 of 23

Set Identification 04RD-2 Name ITEM FILE Form Type MAGT.

Prepared by: HCL

Frequency of Processing DAILY Volume: Average 3000 Peak 4500 Source System I.D. 04

Date 6/15/60

MANAGEMENT RULE NOS.

ELEMENT NAME	ELEMENT CODE	ELEMENT DESCRIPTION					ELEMENT CLASS. (X)						NO. TIMES AN ENTRY MAY APPEAR ON THIS SET		VALIDATION RULE (S)	IDENTIFYING ELEMENT CODES (X)											Ref. Note (#)	REMARKS
		Suffix	Alpha	Numeric	Char. Total	Char. Decimal	Recog.	Ident.	Action	Action Mod.	Info.	Superfluous	Average	Peak		9300100												
STOCK NUMBER	9300100		A	N	07			X					1	1														
SIZE	9144100		A	N	21								1	1		X												
LINE NAME	7941100		A	N	25								1	1		X												
PRICE UN BASE	8664130			N	06	2							1	1		X												
TAX FED EX UN	9464100			N	05	2							1	1		X												

System Identification 04 Process BILLING

INPUT DESCRIPTION

Page 5 of 23

Set Identification 04RD3 et Name SALES TAX FILE Form Type MAGT.

Prepared by: HCD

Frequency of Processing DAILY Volume: Average 120 Peak 200 Source System I. D. 04

Date 6/15/60

MANAGEMENT RULE NOS.

ELEMENT NAME	ELEMENT CODE	ELEMENT DESCRIPTION					ELEMENT CLASS. (X)						NO. TIMES AN ENTRY MAY APPEAR ON THIS SET		VALIDATION RULE (S)	IDENTIFYING ELEMENT CODES (X)												Ref. Note (#)	REMARKS
		Suffix	Alpha	Numeric	Char. Total	Char. Decimal	Recog.	Ident.	Action	Action Mod.	Info.	Superfluous	Average	Peak		9464160												
SALES TAX CODE	9464160		A	N	05			X					1	1															
SALES TAX RC	9464150			N	03	2							1	1		X													

System Identification 04 Process BILLING

INPUT DESCRIPTION

Page 6 of 23

Set Identification 1201 Set Name CUST FILE CHANGE Form Type MANUAL

Prepared by: HCJ

Frequency of Processing WEEKLY Volume: Average 10 Peak 50 Source System I. D. 04

Date 6/15/60

MANAGEMENT RULE NOS. 013

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

Appendix II, Page 10

OUTPUT DESCRIPTION

System Identification 01 Process BILLING Set Name INVOICE Form Type MAT MASTER

Set Identification 1512B Number Copies 8 Volume: Average 500 Peak 1000 Special Form I. D. 1510

Prepared by: HCD

Date 6/15/60

ELEMENT NAME	ELEMENT CODE	ELEMENT DESCRIPTION: Suffix	Alpha	Numeric	Char. Total	Char. Decimal	Element Class. (X): Recog.	Ident.	Other	NO. TIMES AN ENTRY MAY APPEAR ON THIS SET: Average	Peak	SOURCE: Set Type: Process (X)	Retd. (A, B or X)	Action (X)	Source Set Identification for Direct Recording	IDENTIFYING ELEMENT CODES (X): 8368100		9300100									Ref. Note (V)
ACCT CENTER	6017100		A		13				X	1	1			X	1512A	X										
MO CUST ORD	8168620			N	02	0			X	1	1			X	1512A	X										
DA CUST ORD	6836520			N	02	0			X	1	1			X	1512A	X										
YR CUST ORD	9980620			N	02	0			X	1	1			X	1512A	X										
ORDER NO	8368100		A	N	09			X		1	1			X	1512A											
MO SHIPPED	8168650			N	02	0			X	1	1			X	1512A	X										
DA SHIPPED	6836550			N	02	0			X	1	1			X	1512A	X										
YR SHIPPED	9980650			N	02	0			X	1	1			X	1512A	X										
INVOICE NO	7680100		A	N	09				X	1	1			X	1512A	X										
CUST NAME SOLD	6813200		A		70				X	1	1		X		CARD1	X										
CUST ADDR SOLD	6813210		A	N	70				X	1	1		X		CARD1	X										
CUST NAME SHIP	6813220		A		70				X	1	1			X	1512A	X										
CUST ADDR SHIP	6813230		A	N	70				X	1	1			X	1512A	X										
CUST ORDER NO	6813100		A	N	12				X	1	1			X	1512A	X										
SALESMAN NO	8981100			N	05	0			X	1	1			X	1512A	X										
CUST ACCT NO	6813250			N	04	0			X	1	1			X	1512A	X										
TERMS PAYMENT	9496100		A	N	23				X	1	1			X	1512A	X										
SHIPG TERMS	9111100		A		23				X	1	1			X	1512A	X										
CARRIER NAME	6466100		A		24				X	1	1			X	1512A	X										
SHIP FROM TOWN	9110120		A		13				X	1	1			X	1512A	X										
SALES TAX CODE	9464160		A	N	05				X	1	1			X	1512A	X										
PRICE BASE	8664100		A	N	08				X	1	1			X	1512A	X										
SIZE	9144100		A	N	21				X	3	15		X		CARD2	X		X								
LINE NAME	7941100		A	N	25				X	3	15		X		CARD2	X		X								

Appendix II, Page 11

OUTPUT DESCRIPTION

System Identification OA Process BILLING Set Name INVOICE Form Type MAT. MASTER

Set Identification 1512B Number Copies 8 Volume: Average 500 Peak 1000 Special Form I. D. 1510

Prepared by: HCO

Date 6/15/60

ELEMENT NAME	ELEMENT CODE	ELEMENT DESCRIPTION					Element Class. (X)			NO. TIMES AN ENTRY MAY APPEAR ON THIS SET		SOURCE					IDENTIFYING ELEMENT CODES (X)												Ref. Note (V)
		Suffix	Alpha	Numeric	Char. Total	Char. Decimal	Recog.	Ident.	Other	Average	Peak	Set Type: Process (X)	Set Type: Retd. (A, B or X)	Set Type: Action (X)	Source Set Identification for Direct Recording		8968100		9300100										
STOCK NUMBER	9300100		A	N	07			X		3	15			X	1512A		X												
QTY ORDERED	8768500			N	05	0			X	3	15			X	1512A		X		X										
QTY SHIPPED	8768550			N	05	0			X	3	15			X	1512A		X		X										
TAX FED EX UN	9464100			N	05	2			X	3	15		X		OARD2		X		X										
UNIT PRICE	8664110			N	06	2			X	3	15	X					X		X										
DISC PC QSA	6920110			N	02	1			X	1	1			X	1512A		X												
AMT QTY ALLOW	6116120			N	05	2			X	1	1	X					X												
AMT PRCE EXT	6116100			N	07	2			X	3	15	X					X		X										
AMT PRE TAX	6116180			N	08	2			X	1	1	X					X												
AMT FED EX TAX	6116190			N	07	2			X	1	1	X					X												
AMT CUST INV	6116220			N	08	2			X	1	1	X					X												
SALES TAX PC	9464150			N	03	2			X	1	1	X					X												
AMT SALES TAX	6116210			N	04	2			X	1	1	X					X												

OUTPUT DESCRIPTION

System Identification 04 Process BILLING Set Name SALES DATA Form Type MAG. TAPE

Set Identification 0401 Number Copies 1 Volume: Average 2000 Peak 5000 Special Form I. D. _____

Prepared by: HCD

Date 6/15/60

ELEMENT NAME	ELEMENT CODE	ELEMENT DESCRIPTION: Suffix	Alpha	Numeric	Char. Total	Char. Decimal	Element Class. (X): Recog.	Ident.	Other	NO. TIMES AN ENTRY MAY APPEAR ON THIS SET: Average	Peak	SOURCE: Set Type: Process (X)	Retd. (A, B or X)	Action (X)	Source Set Identification for Direct Recording	IDENTIFYING ELEMENT CODES (X): 9300100	Ref. Note (V)
STOCK NUMBER	9300100		A	N	07					1	1			X	1512 A		
MO SHIPPED	8116850			N	02	0				1	1			X	1512 A	X	
DA SHIPPED	6836550			N	02	0				1	1			X	1512 A	X	
YR SHIPPED	9980650			N	02	0				1	1			X	1512 A	X	
SALESMAN NO	8981100			N	05	0				1	1			X	1512 A	X	
CUST ACCT NO	68 13250			N	04	0				1	1			X	1512 A	X	
QTY SHIPPED	8768550			N	05	0				1	1			X	1512 A	X	
AMT PRICE EXT	6116100			N	07	2				1	1	X				X	

OUTPUT DESCRIPTION

System Identification OA ..ocess BILLING Set Name INVENTORY DATA Form Type MAG TAPE

Set Identification OA02 Number Copies 1 Volume: Average 2000 Peak 5000 Special Form I. D. _____

Prepared by: HCD

Date 6/15/60

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

OUTPUT DESCRIPTION

System Identification 040 | Process BILLING | Set Name ORDER ERRORS | Form Type STANDARD PRINT

Set Identification 040301 | Number Copies 2 | Volume: Average 5 | Peak 10 | Special Form I. D. _____

Prepared by: HCD

Date 6/15/60

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

Appendix II, Page 15

OUTPUT DESCRIPTION

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

Set Identification 040302 Number Copies 2 Volume: Average 5 Peak 10 Special Form I. D. _____

Prepared by: HCD

Date 6/15/60

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

Appendix II, Page 16

OUTPUT DESCRIPTION

System Identification 04 Process BILLING Set Name ORDER ERRORS Form Type STANDARD PRINT

Set Identification 040303 Number Copies 2 Volume: Average 20 Peak 40 Special Form I. D. _____

Prepared by: HCD

Date 6/15/60

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

OUTPUT DESCRIPTION

System Identification 04 Process BILLING Set Name ORDER ERRORS Form Type STANDARD PRINT

Set Identification 040304 Number Copies 2 Volume: Average 5 Peak 10 Special Form I. D. _____

Prepared by: HCD

Date 6/15/60

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

Appendix III, Page 18

OUTPUT DESCRIPTION

System Identification 04 . cess BILLING Set Name ORDER ERRORS Form Type STANDARD PRINT

Set Identification 040305 Number Copies 2 Volume: Average 5 Peak 10 Special Form I. D. _____

Prepared by: HCD

Date 6/15/60

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

OUTPUT DESCRIPTION

System Identification 04 Process BILLING Set Name ORDER ERRORS Form Type STANDARD PRINT

Set Identification 040306 Number Copies 2 Volume: Average 2 Peak 5 Special Form I. D. _____

Prepared by: HCD

Date 6/15/60

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

OUTPUT DESCRIPTION

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

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

Prepared by: HCD

Date 6/15/60

ELEMENT NAME	ELEMENT CODE	ELEMENT DESCRIPTION: Suffix	Alpha	Numeric	Char. Total	Char. Decimal	Element Class. (X): Recog.	Ident.	Other	NO. TIMES AN ENTRY MAY APPEAR ON THIS SET: Average	Peak	SOURCE: Set Type: Process (X)	Retd. (A, B or X)	Action (X)	Source Set Identification for Direct Recording	IDENTIFYING ELEMENT CODES (X): 8368100	Ref. Note (✓)
ORDER NO	8368100		A	N	09			X		1	1			X	1512A		
MO SHIPPED	8168650			N	02	0			X	1	1	X				X	
DA SHIPPED	8836550			N	02	0			X	1	1	X				X	
YR SHIPPED	9980650			N	02	0			X	1	1	X				X	

Appendix III, Page 21

OUTPUT DESCRIPTION

System Identification 04 Process BILLING Set Name CUST CHANGE ERROR Form Type STANDARD PRINT

Set Identification 0404 Number Copies 2 Volume: Average 1 Peak 10 Special Form I. D. _____

Prepared by: HCD

Date 6/15/60

ELEMENT NAME	ELEMENT CODE	ELEMENT DESCRIPTION					Element Class. (X)			NO. TIMES AN ENTRY MAY APPEAR ON THIS SET		SOURCE						IDENTIFYING ELEMENT CODES (X)												Ref. Note (V)
		Suffix	Alpha	Numeric	Char. Total	Char. Decimal	Recog.	Ident.	Other	Average	Peak	Set Type: Process (X)	Set Type: Retd. (A, B or X)	Set Type: Action (X)	Source Set Identification for Direct Recording			8981100		6813250										
SALESMAN NO	8981100			N	05	0		X		1	1			X	1201															
CUST ACCT NO	6813250			N	04	0		X		1	1			X	1201			X												
DISC PC SUP BON	6920130			N	02	1			X	1	1	X						X		X										
CUST NAME SOLD	6813200		A		70				X	1	1	X						X		X										
CUST ADDR SOLD	6813210		A	N	70				X	1	1	X						X		X										
POSTING IND	8760100			N	01	0			X	1	1	X						X		X										
ERROR REASON	7000100		A		09				X	1	1	X						X		X										

PROCESS DESCRIPTION

System Identification 04

Process BILLING ☒ Validation

Prepared by HCD

Date 6/15/60

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

Cond: Y = Condition Is Satisfied; N = Condition Is Not Satisfied; Blank = Condition Not Applicable

Action: X = Action to Be Taken; Blank = Action Not to Be Taken

PROCESS DESCRIPTION

System Identification 04

Process BILLING ☒ Validation

Prepared by HCD

Date 6/15/60

LINE NO.	C/A	MANAGEMENT RULE NO. Current	Prior	SOURCE ELEMENT Name, Prior Result or Actual Value	SUF.	Set Ident. or Rule No. for Prior Result	COND. Greater Than	Less Than	Equal To	Oper (+ - / X Σ)	Set = To (V)	SOURCE/DISPOSITION ELEMENT Name, Result, Prior Result or Actual Value	SUF.	Set Ident. or Rule No. for Prior Result	Oper (+ - / X)	Note Ref. (V)	MANAGEMENT RULE SUFFIX AND FREQUENCY A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R
012	C	002		SALES TAX CODE		1512 A			✓			SALES TAX CODE		04RD3		1	Y	N															
																	95	5															
013	A			SALES TAX CODE		1512 A					✓	SALES TAX CODE		040301				X															
014	A			(x b b b b)							✓	SALES TAX CODE		1512 A				X															
015	C	003		SALESMAN NO		1512 A			✓			SALESMAN NO		04RD1		2	Y	N															
016				CUST ACCT NO		1512 A			✓			CUST ACCT NO		04RD1			Y		N														
																	95	2	3														
017	A			SALESMAN NO		1512 A					✓	SALESMAN NO		040302				X	X														
018	A			CUST ACCT NO		1512 A					✓	SALESMAN NO		040302				X	X														
019	C	004		STOCK NO		1512 A			✓			STOCK NO		04RD2		2	Y	N															
																	95	6															
020	A			STOCK NO		1512 A					✓	STOCK NO		040303				X															
021	C	005		QTY SHIPPED		1512 A			✓			/NNNNN/				2	Y	N	Y	Y													
022	C			QTY ORDERED		1512 A			✓			/NNNNN/					Y	Y	N	Y													
023	C			QTY ORDERED		1512 A	✓		✓			QTY SHIPPED		1512 A			Y			N													
																	95	2	2	1													
024	A			QTY SHIPPED		1512 A					✓	QTY SHIPPED		040304				X															
025	A			QTY ORDERED		1512 A					✓	QTY ORDERED		040305					X														
026	A			QTY SHIPPED		1512 A					✓	QTY SHIPPED		040306						X													
027	A			QTY ORDERED		1512 A					✓	QTY ORDERED		040306						X													

MANAGEMENT RULE SUFFIX AND FREQUENCY

Cond: Y = Condition Is Satisfied; N = Condition Is Not Satisfied; Blank = Condition Not Applicable

Action: X = Action to Be Taken; Blank = Action Not to Be Taken

Appendix II, Page 24

PROCESS DESCRIPTION

System Identification OA

Process BILLING ☐ Validation

Prepared by HCD

Date 6/15/60

LINE NO.	C/A	MANAGEMENT RULE NO. Current	MANAGEMENT RULE NO. Prior	SOURCE ELEMENT Name, Prior Result or Actual Value	SOURCE ELEMENT SUF.	SOURCE Set Ident. or Rule No. for Prior Result	COND. Greater Than	COND. Less Than	COND. Equal To	COND. Oper (+ - / X Σ)	Set = To (V)	SOURCE/DISPOSITION ELEMENT Name, Result, Prior Result or Actual Value	SOURCE/DISPOSITION ELEMENT SUF.	SOURCE/DISPOSITION Set Ident. or Rule No. for Prior Result	Oper (+ - / X)	Note Ref. (V)	MANAGEMENT RULE SUFFIX AND FREQUENCY A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R
028	A	006		DISC PC SUP BON		04RD1				X		PRICE UN BASE		04RD2																			
029											✓	UNIT PRICE		1512 B																			
030	C	007	008	DISC PC QSA		1512 A			✓			/BLANK/					Y	N															
																	60	40															
031	A			DISC PC QSA		1512 A				X		AMT PRE TAX		1512 B																			
032											✓	AMT QTY ALLOW		1512 B				X															
033											✓	RESULT A						X															
034				/ZERO/							✓	RESULT A					X																
035	A	008	006	UNIT PRICE		1512 B				X		QTY SHIPPED		1512 A																			
036											✓	AMT PRICE EXT		1512 B																			
037											✓	AMT PRICE EXT		0401																			
038				AMT PRICE EXT						Σ	✓	AMT PRE TAX		1512 B																			
039	A	009		QTY SHIPPED		1512 A				X		TAX FED EX UN		04RD2																			
040											✓	AMT FED EX TAX		1512 B																			
041	A	010	008	AMT PRE TAX		1512 B				+		AMT FED EX TAX		1512 B																			
042			009								✓	RESULT A																					

Cond: Y = Condition Is Satisfied; N = Condition Is Not Satisfied; Blank = Condition Not Applicable

Action: X = Action to Be Taken; Blank = Action Not to Be Taken

PROCESS DESCRIPTION

System Identification 04

Process BILLING ☐ Validation

Prepared by HCD

Date 6/15/60

LINE NO.	C/A	MANAGEMENT RULE NO. Current	MANAGEMENT RULE NO. Prior	SOURCE ELEMENT Name, Prior Result or Actual Value	SUF.	SOURCE Set Ident. or Rule No. for Prior Result	COND. Greater Than	COND. Less Than	COND. Equal To	Oper (+ - / X Σ)	Set = To (V)	SOURCE/DISPOSITION ELEMENT Name, Result, Prior Result or Actual Value	SUF.	SOURCE/DISPOSITION Set Ident. or Rule No. for Prior Result	Oper (+ - / X)	Note Ref. (V)	MANAGEMENT RULE SUFFIX AND FREQUENCY A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R
043	C	011	008	SALES TAX CODE		1512 A			✓			(x b b b b)					Y	N																
																	5	9/5																
044	A			SALES TAX PC		04RD3				X		AMT PRE TAX		1512 B																				
045											✓	AMT SALES TAX		1512 B				X																
046											✓	RESULT A						X																
047				/ZERO/							✓	RESULT A					X																	
048				SALES TAX PC		04RD3					✓	SALES TAX PC		1512 B				X																
049	A	012	011	RESULT A		(010)				-		RESULT A		(007)	+																		
050			010	RESULT A		(011)					✓	AMT CUST INV		1512 B																				
			007																															

MANAGEMENT RULE SUFFIX AND FREQUENCY

Cond: Y = Condition Is Satisfied
N = Condition Is Not Satisfied
Blank = Condition Not Applicable

Action: X = Action to Be Taken
Blank = Action Not to Be Taken

PROCESS DESCRIPTION

System Identification 04

Process BILLING ☐ Validation

Prepared by HCD

Date 6/15/60

LINE NO.	C/A	MANAGEMENT RULE NO. Current	MANAGEMENT RULE NO. Prior	SOURCE ELEMENT Name, Prior Result or Actual Value	SUF.	SOURCE Set Ident. or Rule No. for Prior Result	COND. Greater Than	Less Than	Equal To	Oper (+ - / X Σ)	Set = To (V)	SOURCE/DISPOSITION ELEMENT Name, Result, Prior Result or Actual Value	SUF.	SOURCE/DISPOSITION Set Ident. or Rule No. for Prior Result	Oper (+ - / X)	Note Ref. (V)	MANAGEMENT RULE SUFFIX AND FREQUENCY A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R
051	C	013		SALESMAN NO		1201			✓			SALESMAN NO		04RD1			Y	Y	Y	N		N		N										
052				CUST ACCT NO		1201			✓			CUST ACCT NO		04RD1			Y	Y	Y		N		N		N									
053				POSTING IND		1201			✓			(1)					Y			Y	Y													
054									✓			(2)						Y				Y	Y											
055									✓			(3)							Y					Y	Y								
																	1/5	1	5/0	1	1	1/5	1/5	1	1								
056	A			/BLANK/							✓			04RD1			X																
057						1201					✓			04RD1								X	X										
058				DISC PC SUP BON		1201					✓	DISC PC SUP BON		04RD1					X														
059				DISC PC SUP BON		1201					✓	DISC PC SUP BON		0404				X		X	X			X	X								
060				CUST NAME SOLD		1201					✓	CUST NAME SOLD		0404				X		X	X												
061				CUST ADDR SOLD		1201					✓	CUST ADDR SOLD		0404				X		X	X												
062				(MATCHED)							✓	ERROR REASON		0404				X															
063				(UNMATCHED)							✓	ERROR REASON		0404						X	X			X	X								

MANAGEMENT RULE SUFFIX AND FREQUENCY

Cond: Y = Condition Is Satisfied
N = Condition Is Not Satisfied
Blank = Condition Not Applicable

Action: X = Action to Be Taken
Blank = Action Not to Be Taken

STANDARD REFERENCE NOTES FOR VALIDATIONS

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