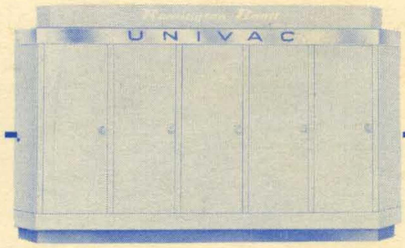
U1518

UNIVAC® FLOW-MATIC PROGRAMMING SYSTEM

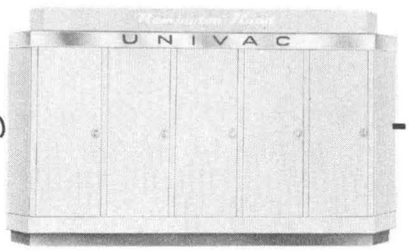
UNIVAC®

- - COMPARE PRODUCT NO. (A) - -  - B00940 - - - L00880 -

FLOW-MATIC PROGRAMMING SYSTEM

Remington Rand Univac
DIVISION OF SPERRY RAND CORPORATION

UNIVAC®



- - - - -COMPARE PRODUCT NO. (A) - - - - B00940 - - - L00880 - - - -

FLOW-MATIC PROGRAMMING SYSTEM

ANOTHER SERVICE OF . . .

MANAGEMENT SERVICES AND OPERATIONS RESEARCH DEPARTMENT

Remington Rand Univac

DIVISION OF SPERRY RAND CORPORATION

Preface

WHAT IS THE UNIVAC FLOW-MATIC PROGRAMMING SYSTEM?

THE UNIVAC FLOW-MATIC SYSTEM is a revolutionary new programming aid developed for the UNIVAC Data Automation System by the Automatic Programming Development Group of the Remington Rand Division, Sperry Rand Corporation. Using an English language description of application requirements as its instruction code, this new product is especially designed for use by those who know and can best define their data processing needs. With this new system, the Computer is directed to accept descriptions of application requirements in the businessman's vocabulary and translate these descriptions automatically and accurately into detailed coded instructions.

WHO CAN USE THE UNIVAC FLOW-MATIC SYSTEM?

The systems and procedures analysts, the accountants, operating management can use the *UNIVAC FLOW-MATIC SYSTEM* with little training. Familiarity with detailed computer coding is not necessary. Familiarity with the desired system, and the ability to describe it are the only prerequisites. The experienced programmer will also find that the *UNIVAC FLOW-MATIC SYSTEM* facilitates program preparation.

WHY USE UNIVAC FLOW-MATIC?

The *UNIVAC FLOW-MATIC AUTOMATIC PROGRAMMING SYSTEM* offers to its user unprecedented benefits, -- benefits such as:

Faster and more accurate Programming - The coding process utilized reduces the elapsed time between the original conception of an application and the checked out final processing. The concise method of expression greatly reduces chance of error.

Easier Programming Analysis - The inherent step by step approach leaves a trail of easily understood documentation, important in retrospect if it becomes necessary to alter the programs either because of changed requirements or modifications in the data processing system.

Checks Systems Design - The ease of preparing *UNIVAC FLOW-MATIC* charts, plus the flexibility with which changes can be made facilitates greater use of pilot systems to check the basic logic of the system design.

One-Shot Jobs Now Practical - The programming of many one-shot jobs formerly considered impractical is now not only feasible and economical, but also provides invaluable additional fact power to decision making management.

Flexibility - Changes in either the processing procedure or data format can be accomplished independently without affecting the other.

ABOUT THIS MANUAL

Straight-forward in approach, this manual is so designed to serve the gradational need of the user. Later chapters and appendices furnish detailed information for the experienced programmer.

Table Of Contents

Chapter		*Page*
	INTRODUCTION	
1	A COMPLETE SYSTEM WHICH MEETS ALL USER PROCESSING REQUIREMENTS	1
	AVAILABLE EQUIPMENTS AND THEIR FUNCTIONS	2
	THE UNIVAC II CENTRAL COMPUTER	2
	THE UNIVAC II CONTROL GROUP	2
	UNIVAC INPUT DEVICES	4
	UNIVAC INPUT-OUTPUT DEVICES	8
	UNIVAC OUTPUT DEVICES	8
	ILLUSTRATION OF INTERRELATIONSHIP OF EQUIPMENT	12
2	THE FLOW-MATIC METHOD *The steps to be followed in applying this new method of programming are described in terms of a data-processing problem.*	13
3	FLOW-MATIC CHARTING AND PROGRAM-WRITING *The procedures for drawing flow charts and writing English-language programs are described. Sample Problem 1 is programmed.*	26
4	FILE-DATA LAYOUT AND DESIGN *The methods of writing descriptions of data files, items and fields is given, along with the pre-printed forms used.*	35
5	INTERMEDIATE OR WORKING STORAGE *Sample Problem 2 is introduced requiring working storage. The method of utilizing this feature of FLOW-MATIC is described.*	60
6	RELATIVE MACHINE CODING, X-1 *In order that special or unique subroutines may be included in the compiled program, a method of machine coding is shown. Sample Problem 3 indicates how this is used in a program.*	72
7	THE FLOW-MATIC COMPILING ROUTINE *A rudimentary description is given of how the compiler operates, and of the record of compilation prepared for the user.*	83
Appendix		
A	FLOW-MATIC FUNCTIONS, GUIDE FOR WRITING FLOW-MATIC CODE, FLOW-MATIC STATEMENTS	91
B	DATA DESIGN PRE-PRINTED FORMS	101
C	RELATIVE MACHINE CODING X-1	105
D	FLOW-MATIC OPERATING INSTRUCTIONS	111
	INDEX	115

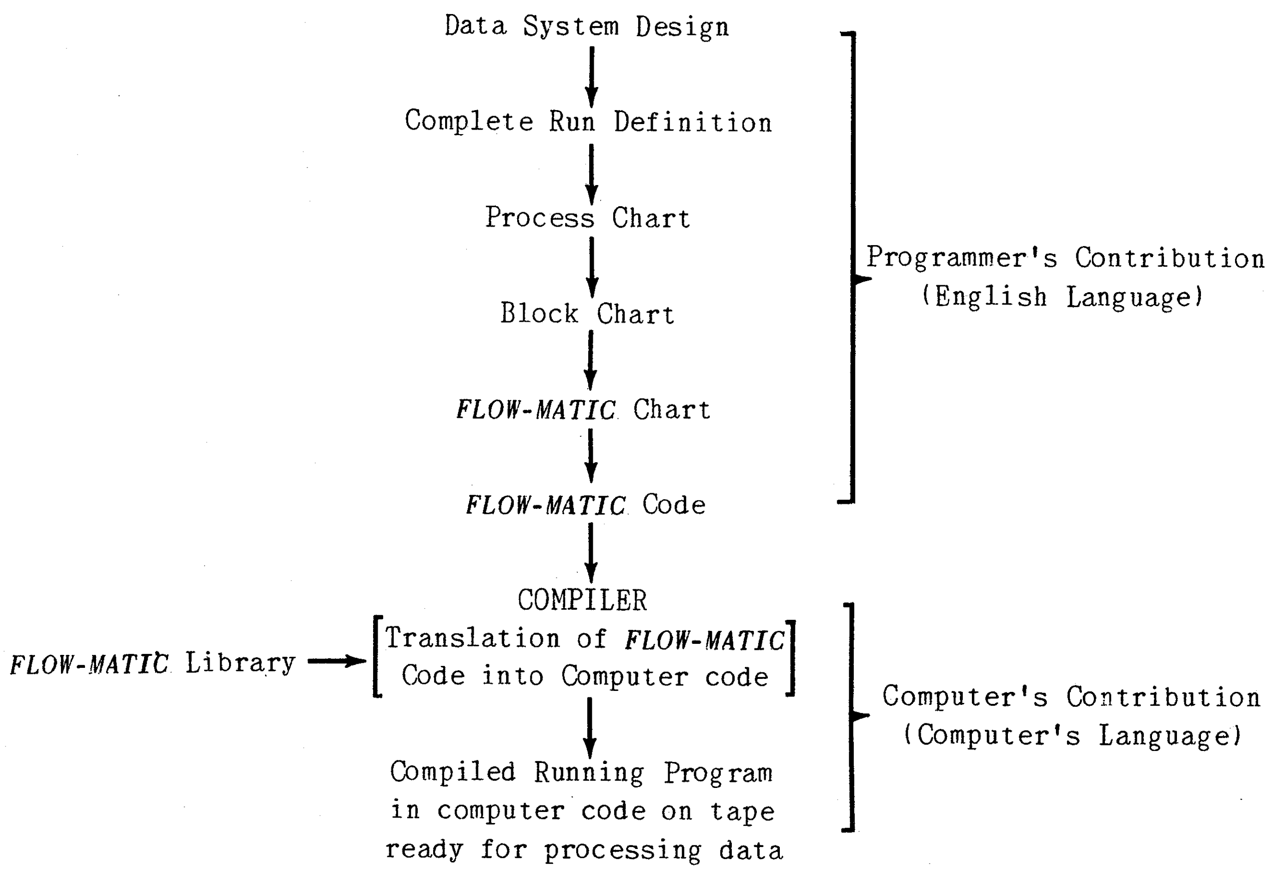
Introduction

The *FLOW-MATIC* programming system provides for the UNIVAC user an entirely new, different, versatile method of writing programs. The key development in this system is the conversion of the *FLOW-MATIC* code in the businessman's vocabulary into standard computer code. The primary objective in the development of this system has been to create a tool to assist analysts in the preparation of programs for data-processing systems. This is the first major step in the direction of a general-purpose programming aid for data-processing users.

The use of the *FLOW-MATIC* System involves the following steps:

$$\begin{tikzcd}
\text{Data System Design} \arrow[d] \\
\text{Complete Run Definition} \arrow[d] \\
\text{Process Chart of the Run} \arrow[d] \\
\text{Block Chart of the Program} \arrow[d] \\
\textit{FLOW-MATIC}\text{ Chart} \arrow[d] \\
\textit{FLOW-MATIC}\text{ Code}
\end{tikzcd}$$

The user may also be interested in the general picture of the entire process in preparing a program with the aid of the *FLOW-MATIC* compiling routine as shown in the graphic presentation as follows:


Data System Design
Complete Run Definition
Process Chart
Block Chart
FLOW-MATIC Chart
FLOW-MATIC Code
Programmer's Contribution (English Language)
COMPILER
Translation of *FLOW-MATIC* Code into Computer code
FLOW-MATIC Library
Compiled Running Program in computer code on tape ready for processing data
Computer's Contribution (Computer's Language)

The primary consideration in the design of the *FLOW-MATIC* system is the (data processing) user and the types of programs required by the system he defines. It provides a complete method, or procedure, which begins with the first definition of the problem or application, and continues throughout the checking out phase and into the program and system refinements which follow the initial operation of the UNIVAC system.

Both the manner and the extent to which the *FLOW-MATIC* System is used depend on the individual who uses it. The expert programmer may use the *FLOW-MATIC* procedure during the initial definition of UNIVAC runs to facilitate communication between the computer programming group and operating management. If the major function of the expert programmer is to supervise a number of less experienced people, he can spend maximum time in the definition and analysis phases, carrying out the steps of the *FLOW-MATIC* procedure only so far as is necessary, depending upon the varying abilities of his individual programmers. Still another possibility is that of utilizing *FLOW-MATIC* to produce the first draft of a program which, because of its high degree of repetition, may later require ingenious modification and application of the "tricks of the trade" known to the expert programmers.

One advantage the *FLOW-MATIC* programming system offers to a data-processing organization in its early stages of growth is the reduction of personnel training time. Within a comparatively short period they will become expert *FLOW-MATIC* programmers and system designers. As with any powerful, general-purpose tool, users increase in competence as they gain experience. This does not, of course, remove the need for experienced, career programmers. Every UNIVAC Data Automation System installation requires, and is benefited by, experienced career programmers who discover operational flaws in programs and systems, correct them, utilize their intensive knowledge to solve especially difficult programming problems, and add to the effectiveness of *FLOW-MATIC* itself.

It is important to point out that *FLOW-MATIC* is a growing system designed for continued expansion and development in a variety of directions, all within the basic framework of the existing structure. It is expected that this growth will proceed in such a way as to make the system increasingly useful.

Although the programmer who uses the *FLOW-MATIC* Compiling system is not required to know the internal operation of the UNIVAC Data Automation System, he must be familiar with the manner in which the computer reads input data, performs various operations and delivers the desired results. Chapter I gives a brief description of the UNIVAC Data Automation System together with the specific function of each individual piece of equipment.

Chapter II provides the reader with a quick, comprehensive view of the *FLOW-MATIC* method. It presents a sample application, and traces the procedure to be followed in programming with this system. Later chapters develop the detailed techniques involved in each step of the procedure, and through the introduction of extensions of the basic application, additional features of *FLOW-MATIC* are explained.

chapter 1

A Complete System Which Meets All User Processing Requirements

The Univac II Data Automation System is a complete and well balanced data processing system. It will accept and prepare information through a wide variety of standard data-recording media. The user gains versatility most economically since the Central Computer, that unit which performs the actual processing, can read and write information directly through the magnetic tape which is one of the most rapid input-output media in use today. Peripheral equipments convert all recorded data into the form acceptable to the Central Computer, or from the form prepared by the Central Computer (Univac II System code on magnetic tape) to the desired form. In this way, the system has a dual advantage. First, the Central Computer need not be hampered in its processing task by the necessity of working directly with input-output media unworthy of its lightning-fast internal operating speeds. Secondly, the Central Computer need not be involved in conversion process which can most economically be handled by peripheral equipments on an off-line basis.

From a wide variety of available equipments each Univac II System user chooses the units which, when molded into a system, best meet his overall data processing requirements.

AVAILABLE EQUIPMENTS AND THEIR FUNCTIONS

THE UNIVAC II CENTRAL COMPUTER

The Univac II Central Computer in Figure 1 is the heart of the Univac II Data Automation System. It performs all arithmetic and logical operations.

In the execution of a typical data processing task the Central Computer performs the following basic operations:

1. Step-by-step instructions, stating specifically the operations to be performed on the data, are read by the Central Computer from magnetic tape and stored internally within the Central Computer. Obeying the stored instructions, the Central Computer then automatically .....
2. Reads the data from magnetic tape and stores it internally.
3. Performs all operations upon the data indicated by the instructions, and stores the results internally.
4. Reads the results from storage and writes them on magnetic tape.

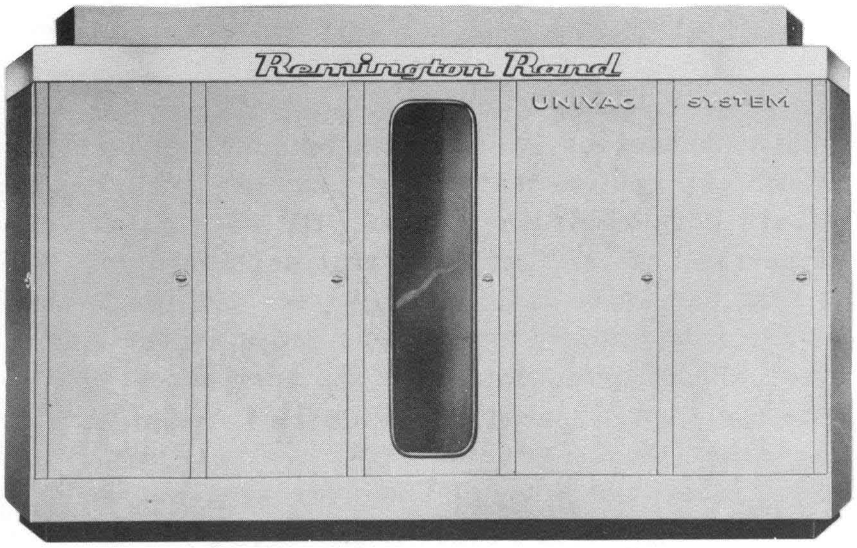


FIGURE 1

All operations are self-checked to ensure that they are performed with the unwavering accuracy and dependability that has become associated with the name UNIVAC.

THE UNIVAC II CONTROL GROUP

Two control units are directly connected to the Central Computer, and each in its own way, provides some indication of the actions of the Central Computer.

Univac Supervisory Control Console

The Univac Supervisory Control Console (Figure 2) provides the operator with a continuous picture of the operations taking place within the Central Computer. It also provides visual indication whenever an error occurs in any operation, identifying the faulty circuit for the maintenance technician.

Although the Central Computer is designed to operate automatically, there are occasions when manual intervention may be desirable. The Univac Supervisory Control Console includes a keyboard by means of which the operator can type information directly into the Central Computer. A group of switches and buttons on the Console allows the interruption of automatic operations and the institution of changes in their course or the substitution or insertion of other operations.

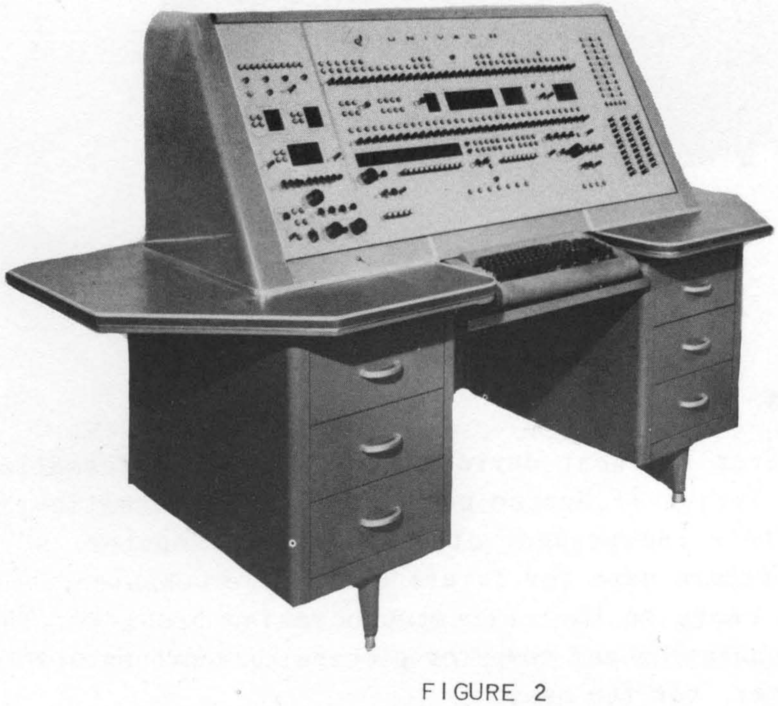


FIGURE 2

Univac Supervisory Control Printer

The Univac Supervisory Control Printer (Figure 3) is a modified electric typewriter which prints information directly from the Central Computer. Its primary function is to provide the operator, in easily readable form, information concerning the processing being performed within the Central Computer. This unit is sometimes employed for printing processing results; however, it is used for this purpose only when the information to be printed is not lengthy.

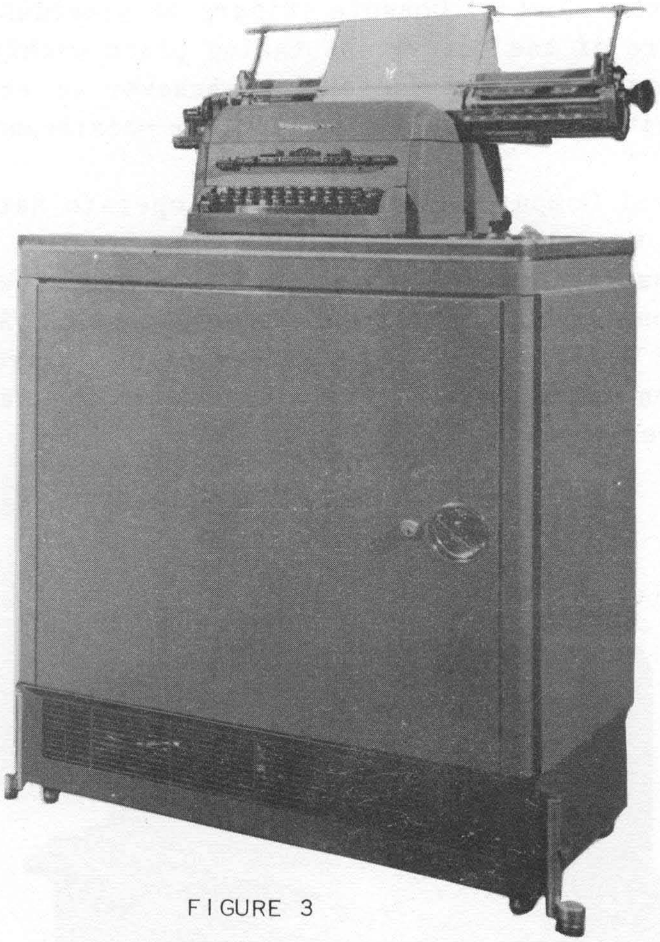


FIGURE 3

UNIVAC INPUT DEVICES

The function of Univac II input devices is to convert information from its original form into Univac II System code recorded on magnetic tape. These devices are completely independent of the Central Computer, so that while the input devices prepare data for future use by the computer, the computer, itself, is free to carry on the current processing problems. This ability to overlap input preparation and computer processing represents a large saving of time and thus money, for the user.

Univac Unityper II

The Univac Unityper II (Figure 4) is a device by means of which information legible to its human operator can be recorded on magnetic tape. This device is somewhat larger than, though similar in appearance to, an electric typewriter. The 26 letters of the alphabet, 10 numerals, and some special Univac II System Symbols are represented on the keyboard of this device in an array similar to the familiar typewriter keyboard pattern. Striking a Unityper II key causes:

1. A pattern of magnetic spots representing the Univac II System Code for the character represented on the key to be recorded on a magnetic tape mounted in the upper portion of the device, and
2. That character to be printed on a piece of copy paper mounted on the carriage.

Thus, recording information on magnetic tape with the Unityper II involves little more than a retyping of the information. Information is tape-recorded by the Unityper II at a density of 50 characters per inch with a 2.4 inch spacing between each consecutive 120 characters.

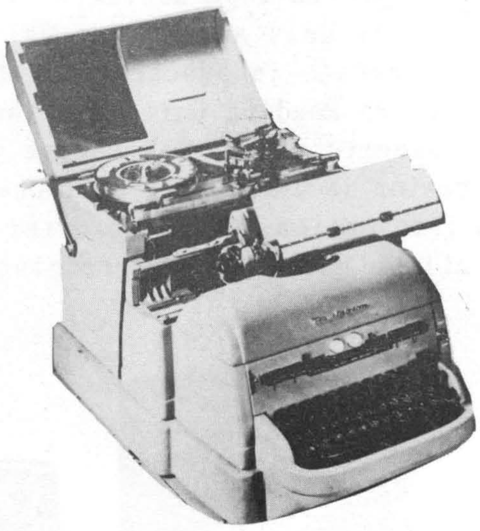


FIGURE 4

Univac Verifier

The Univac Verifier (Figure 5) is a unit of peripheral equipment which can operate in any one of two capacities:

1. As a primary input device which records information on magnetic tape by means of a typewriter keyboard in very much the same manner as the Unityper II.

2. As a proof reading device which corroborates information recorded on tape and permits the correction of detected errors.

Its primary use is as a proof reading and correcting device. Information is recorded by the Verifier at a recording density of 50 characters per inch with a 2.4 inch spacing between each 120 characters.

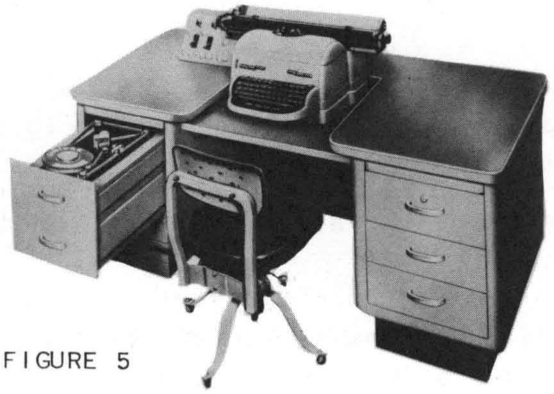


FIGURE 5

Univac Punched Card-to-Magnetic Tape Converter

The Univac Punched Card-to-Magnetic Tape Converter (Figure 6) consisting of a card Reading Unit, a Control Unit and a Tape Unit, allows the entry of information into the Univac System in punched card form. Cards are loaded into the intake bin of the Card Reading Unit, and the information read from the cards is recorded on magnetic tape. The entire process is accomplished automatically and its operation is completely self-checked to ensure complete accuracy of the recorded information. The Converter is equipped with a removable plugboard which allows automatic rearranging of information during the conversion process.

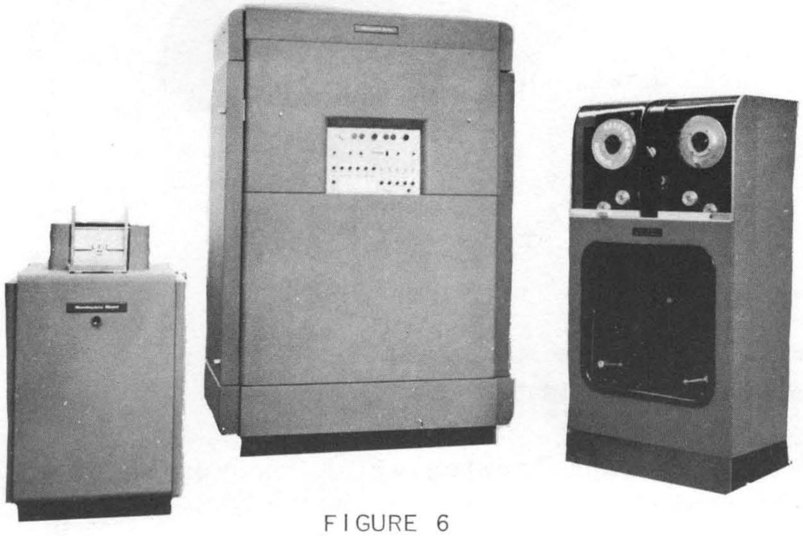


FIGURE 6

Univac Punched Card-to-Magnetic Tape Converters are offered in two models. One handles standard 90-column punched cards; the other handles standard 80-column punched cards. Both models operate at a maximum conversion rate of 240 cards per minute and record information at a density of 128 characters per inch with a 1.8 inch space between each 120 characters, and a 2.4 inch space between each 720 characters.

Univac Paper Tape-to-Magnetic Tape Converter

The Univac Paper Tape-to-Magnetic Tape Converter (Figure 7) is a device consisting of a Perforated Tape Reader, a Translator and Control Unit, and a Magnetic Tape Recorder. This equipment allows information recorded on paper tape to be entered directly into the Univac II System. Reels or message lengths of punched paper tape generated by teletypewriters, automatic typewriters, adding or bookkeeping machines with tape punchers attached, and punched card to perforated tape converters may be mounted on the Tape Reader. Information contained in tapes are automatically translated into Univac II System Code and recorded on magnetic tape. Deletion of certain punched paper tape symbols, and addition of some Univac II System Symbols may be accomplished automatically during the conversion process. The entire operation is completely self-checked to ensure complete accuracy of the conversion process. The Univac Paper Tape-to-Magnetic Tape Converter operates at a maximum conversion rate of 200 characters per second and records information at a density of 128 characters per inch, placing a one inch space between each 120 characters, and a 2.4 inch space between each 720 characters.

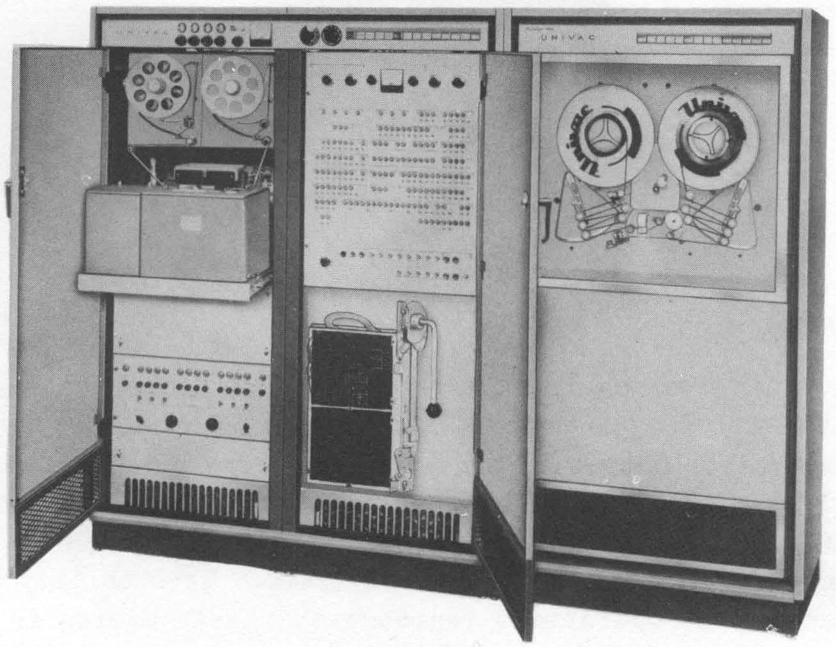


FIGURE 7

UNIVAC INPUT-OUTPUT DEVICES

Univac Uniservo

The Univac Uniservo (Figure 8) is the device through which the Central Computer communicates with its magnetic tapes. A maximum of 16 Uniservos may be directly connected to the Univac II Central Computer. Each Uniservo contains a "read-write" head and mechanism for moving the magnetic tape past the head at a speed of 100 inches per second. Each Uniservo is capable of reading tape moving in the forward direction, reading tape moving in the backward direction, writing on tape moving in the forward direction, and rewinding its tape. Reading from any one Uniservo, writing on any other Uniservo, rewinding the tape on any number of the remaining Uniservos may be carried on simultaneously with Central Computer processing. Uniservo operations are controlled by the Central Computer through programmed instructions.

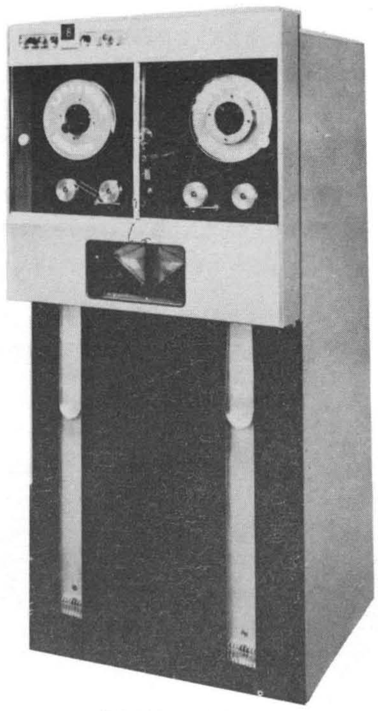


FIGURE 8

UNIVAC OUTPUT DEVICES

Univac II output devices allow the system to prepare processed results in a wide variety of forms. They automatically convert information contained on tapes produced by Central Computer processing into the desired form. All of these output devices operate with complete independence of the Central Computer. Thus, the computer is free to handle further processing while the results of the previous problem are being converted. This ability to overlap conversion and processing operations represents a great saving in time, and money for the user.

Univac Uniprinter

The Univac Uniprinter (Figure 9) consists of a Tape Reader and a Printing Unit which is a modified electric typewriter. A reel of magnetic tape, containing the information to be printed, is mounted on the Tape Reader. As information is read from the tape, it is printed by the electric typewriter. The Univac Uniprinter, which accepts tapes recorded at 25 characters per inch, prints at a rate of 10 characters per second, and is usually used for low volume output printing, such as the preparation of management reports.

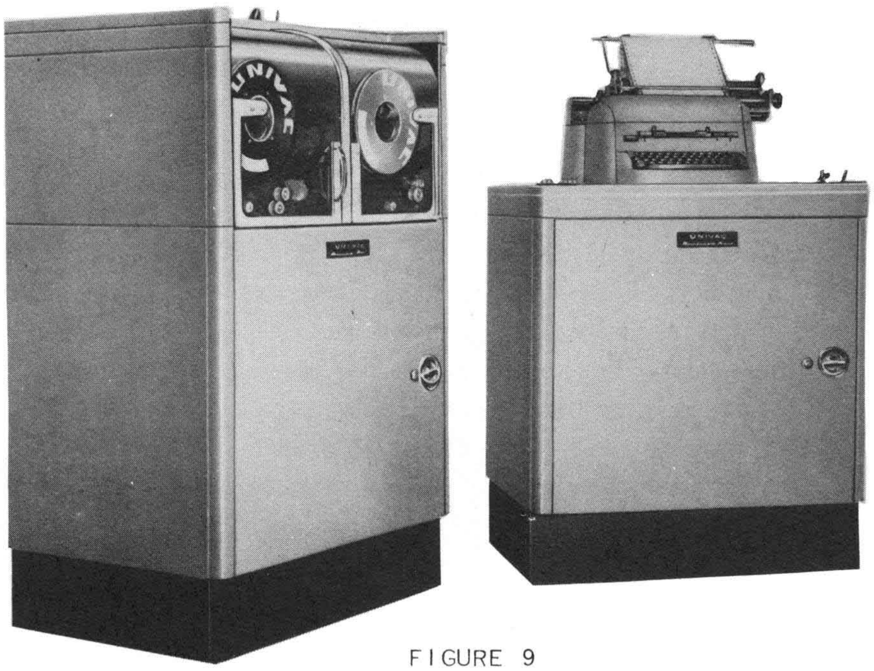


FIGURE 9

Univac High-Speed Printer

The Univac High-Speed Printer (Figure 10) is used for large volume printing. This four unit assembly, consisting of a Tape Reader, a Storage Unit, a control Unit, and a Printer, reads magnetic tape and converts the information recorded thereon into printed copy. The High-Speed Printer prints an entire line at a time. Each line may contain as many as 130 characters, and printing is accomplished at a maximum rate of 600 lines per minute. A removable plug-board mounted in the Control Unit controls the format of the printed page and affords wide flexibility in the arrangement of the printed information, reducing the editing and thus the processing time required of the Central Computer. The entire operation of this device is completely self-checked to ensure that each character printed is the exact one recorded on the magnetic tape. It accepts information tape-recorded at a density of from 50 to 128 characters per inch with at least one inch space between each 120 characters.

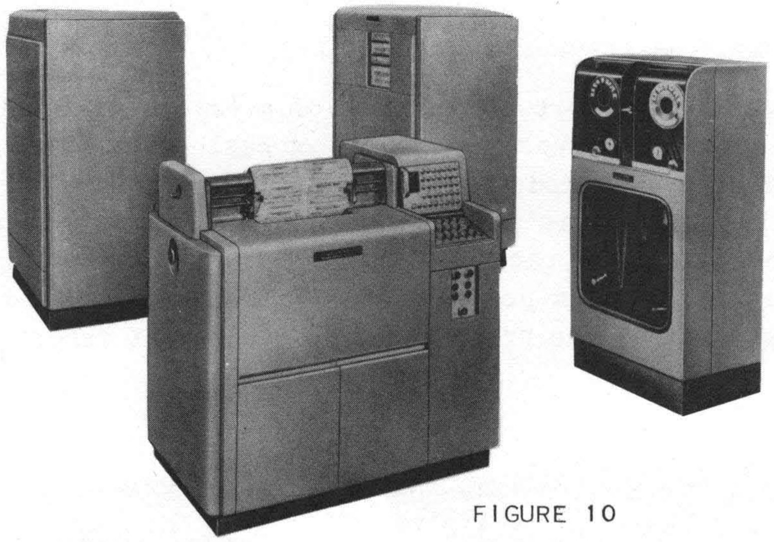


FIGURE 10

Univac Magnetic Tape-to-Card Converter

The Univac Magnetic Tape-to-Card Converter (Figure 11) consists of three units: a Tape Unit, a Card Punch Unit and an Electronic Cabinet containing the circuitry necessary to control and check the Tape and Card Punch Units. This piece of equipment reads information from magnetic tape and converts the information into standard punched cards at a rate of 120 cards per minute. A removable plugboard permits the selection and rearrangement of information during the conversion process. The Univac Magnetic Tape-to-card Converter accepts information tape-recorded at a density of 128 characters per inch with at least one tenth inch space between each 120 characters and 2.4 inch space between each 720 characters. Its entire operation is completely self-checked to ensure proper conversion.

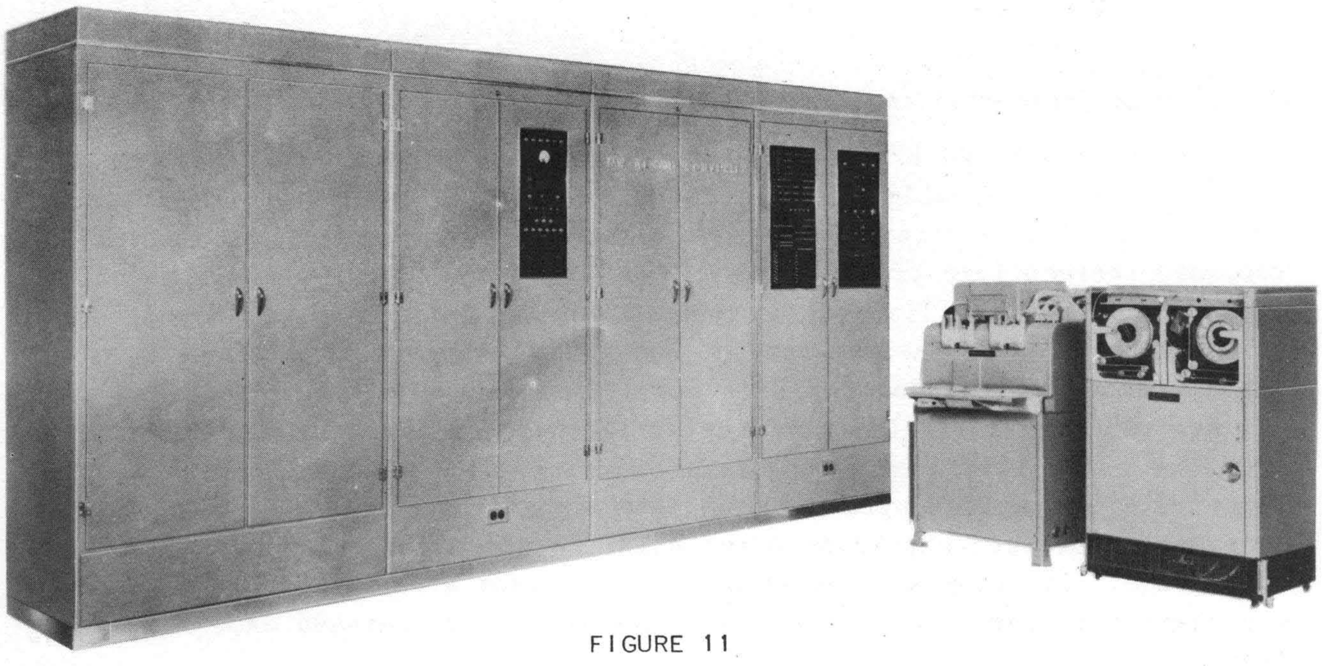


FIGURE 11

Univac Magnetic Tape-to-Paper Tape Converter

The Univac Magnetic Tape-to-Paper Tape Converter (Figure 12) consists of a Magnetic Tape Unit, a Translator and Control Unit, and a Paper Tape Punch. It punches information recorded on magnetic tape into paper tape. The punched paper tapes may then be used directly to send information via a teletypewriter.

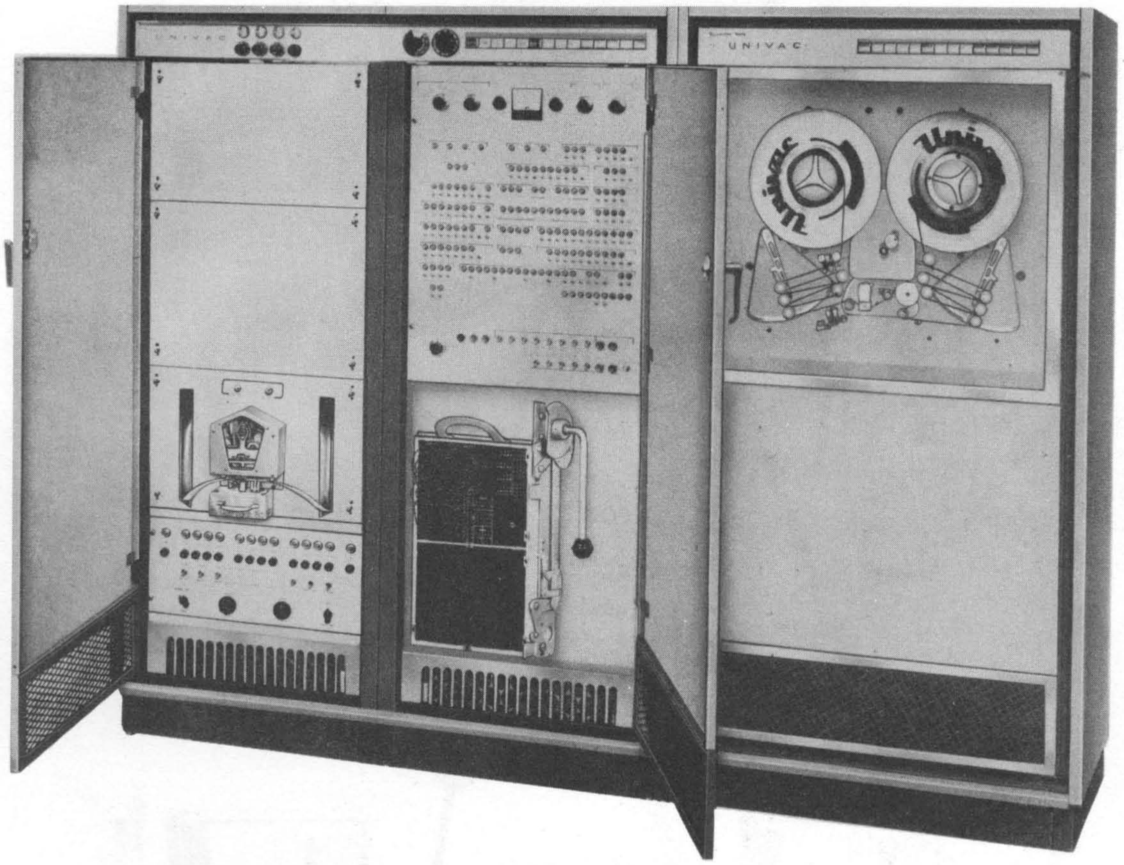


FIGURE 12

As with all Univac II equipment the operation of the Magnetic Tape-to-Paper Tape Converter is completely self-checked to ensure accurate conversion. This conversion is accomplished at a maximum rate of 60 characters per second. It accepts information recorded at a density of 128 characters per inch with at least a 1 inch space between each 120 characters.

INFORMATION CAN ENTER AND EMERGE FROM THE UNIVAC II SYSTEM THROUGH A VARIETY OF STANDARD DATA RECORDING MEDIA

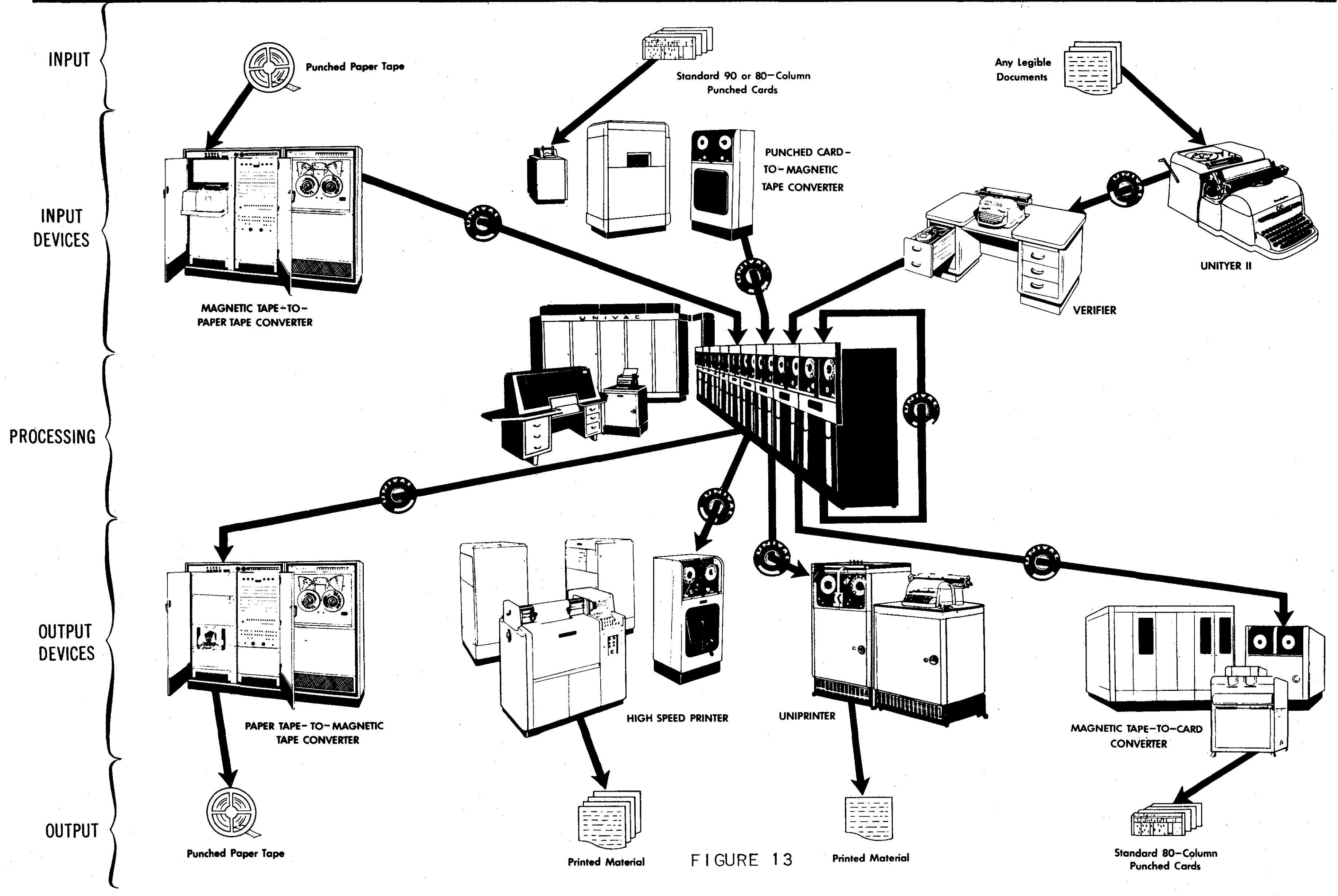

INPUT
INPUT DEVICES
PROCESSING
OUTPUT DEVICES
OUTPUT
Punched Paper Tape
Standard 90 or 80–Column Punched Cards
Any Legible Documents
MAGNETIC TAPE-TO-PAPER TAPE CONVERTER
PUNCHED CARD-TO-MAGNETIC TAPE CONVERTER
VERIFIER
UNITYER II
UNIVAC
PAPER TAPE-TO-MAGNETIC TAPE CONVERTER
HIGH SPEED PRINTER
UNIPRINTER
MAGNETIC TAPE-TO-CARD CONVERTER
Punched Paper Tape
Printed Material
Printed Material
Standard 80–Column Punched Cards

FIGURE 13

chapter 2

The Flow-Matic Method

In the introduction it was stated that the *FLOW-MATIC* System provides an entirely new method of programming. In order to demonstrate this method, a simple data processing run is developed through the steps necessary to prepare the computer coded program. The run prepared is representative of a large class of business data processing runs. Typically, a series of such runs linked together form a data processing system which, depending on the subject matter of the programs, computes a payroll, adjusts inventory, performs sales accounting, or prepares labor distribution.

Each run in any such system is characterized by a flow into the computer of files of data, read from tapes mounted on one or more input Uniservos, and a flow out from the computer of data written on tapes, also mounted on one or more Uniservos. The function of the *FLOW-MATIC* programming system is to produce the program which controls the flow of data through UNIVAC and performs the required processing of the data.

A file stored on magnetic tape is recorded with identifying information written at the beginning of the tape, and sentinels indicating the end of the data also entered on the tape. The body of the file contains a varying number of file items, each representing a separate entry in the file and usually identified by some key information such as payroll number, customer name, or stock number. Frequently the items in a file are in ascending sequence according to this key. Each item within a file contains data elements related to the item, arranged in a systematic format. These data fields will be consistently placed within all items in the file. Thus, for example, by scanning the same relative position in the items of a typical payroll file, the pay rates for employees can be located.

With these few notions of the manner in which business data is stored on magnetic tape, the process chart or run chart can be considered. The process chart is the most general picture of a UNIVAC data processing system. Figure 14 shows a series of tape files represented by circles, linked by a series of boxes which represent computer operations or runs.

SAMPLE PROCESS CHART

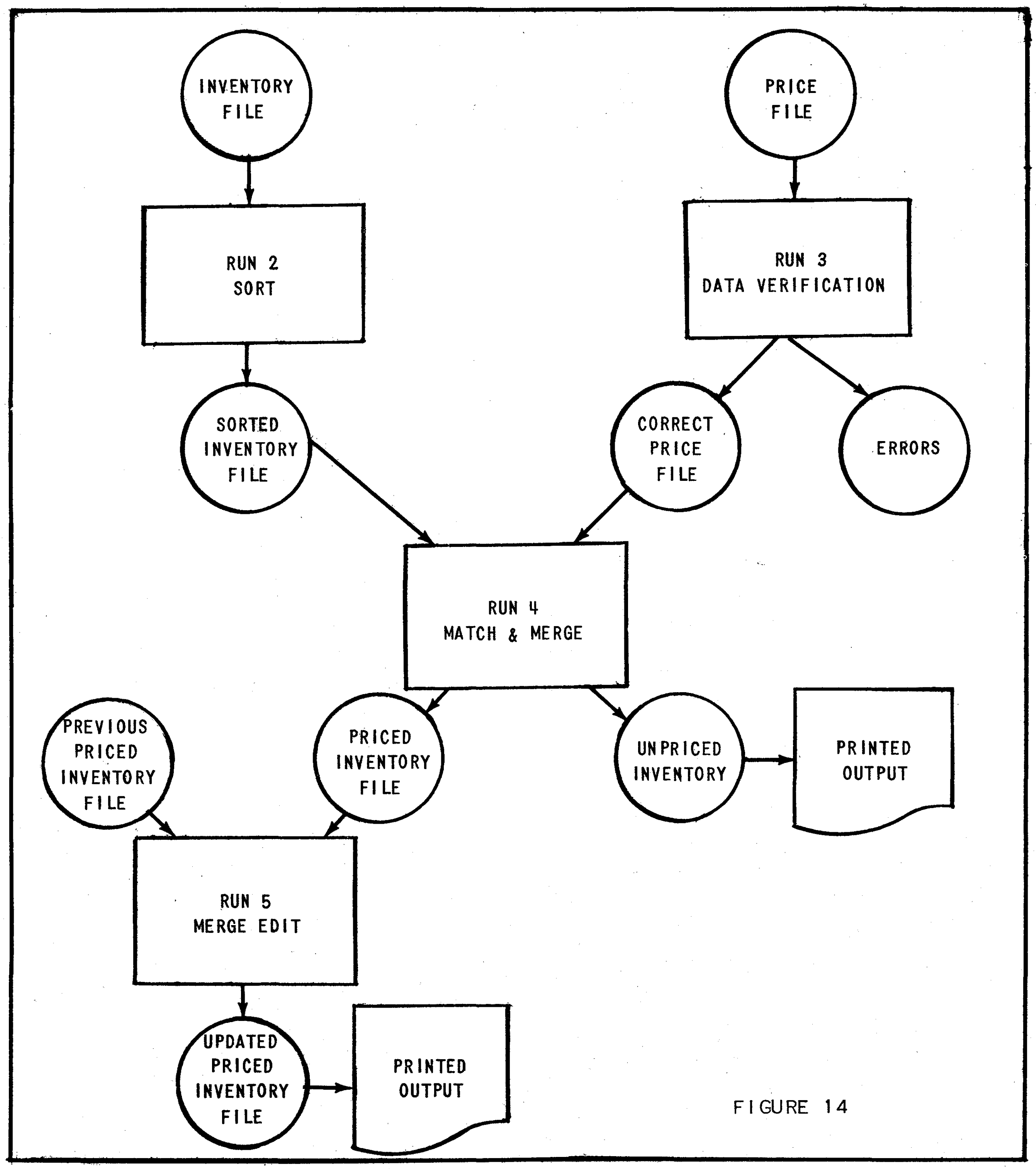


FIGURE 14

Figure 14 is a process chart which describes part of a system of runs. Prior to the preparation of such a chart, a comprehensive study of the system must be made. Some of the points to be considered in this study are the files to be processed, the computer operations required, and the information desired as output.

When the process chart has been prepared, the programming task begins. This is the starting point of the *FLOW-MATIC* method. Consider, therefore, the function of run number 4 of the process chart (Figure 15).

PROCESS CHART
ABC MANUFACTURING COMPANY INVENTORY

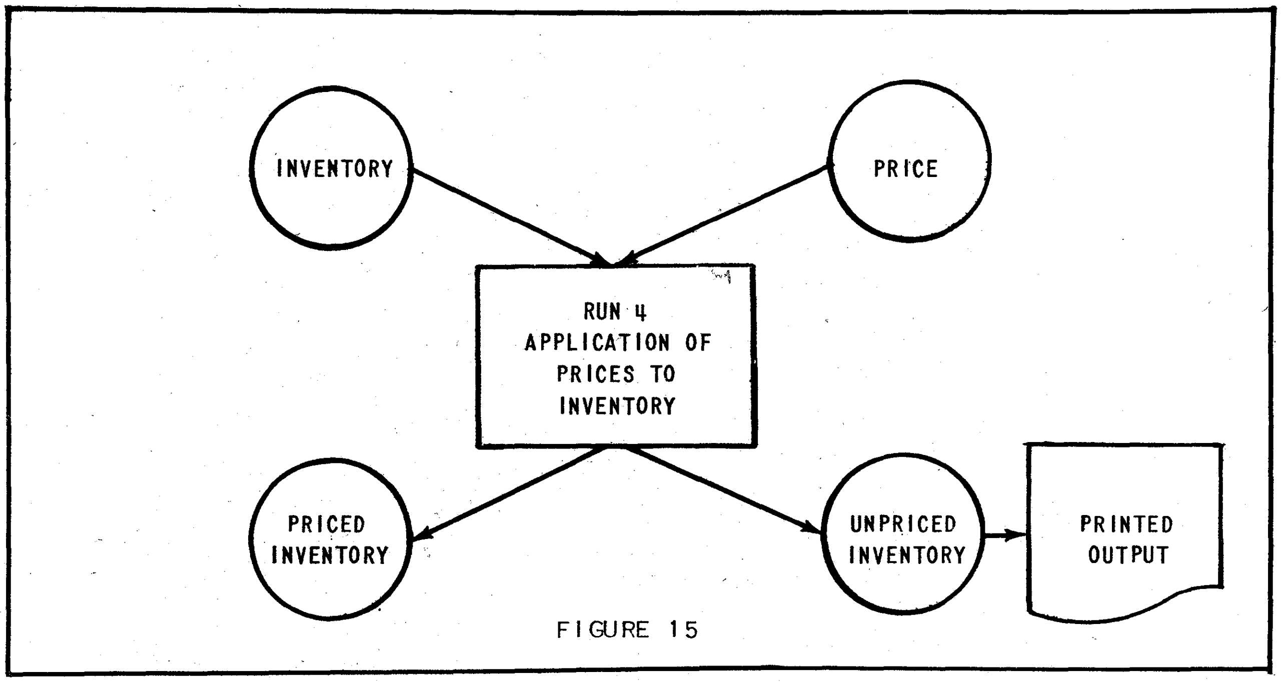
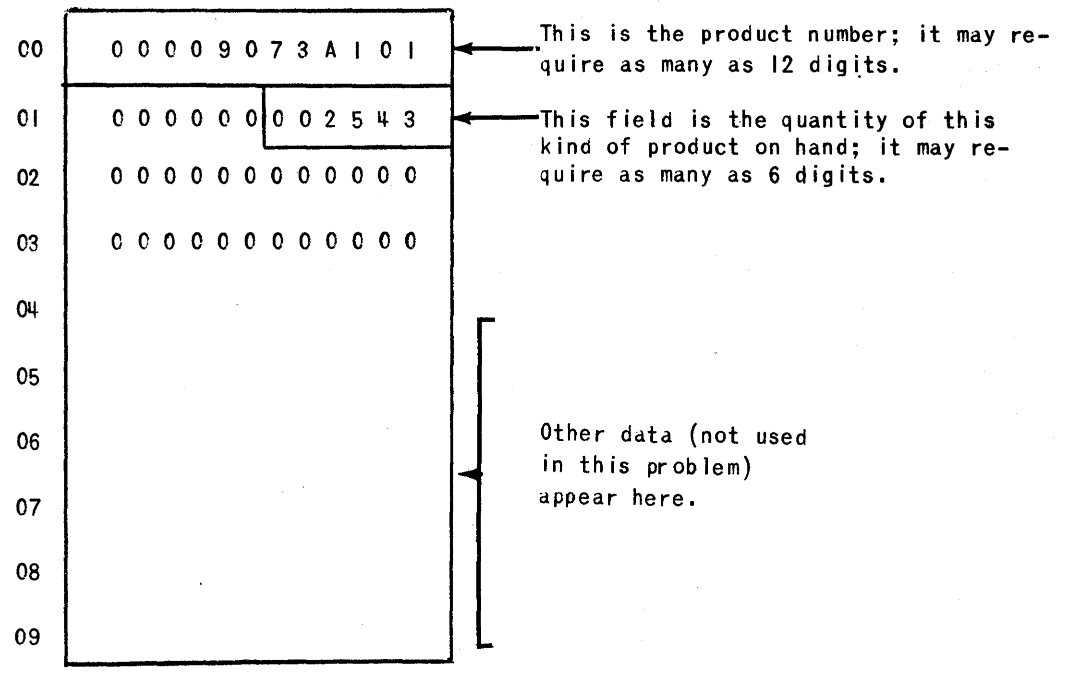


FIGURE 15

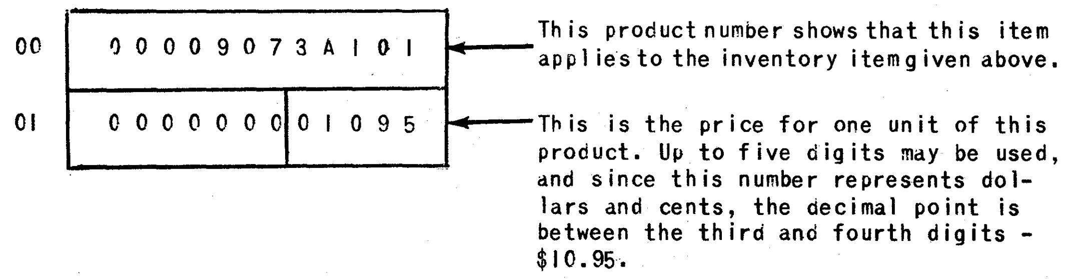
Suppose run 4 is part of the UNIVAC inventory system for the ABC Manufacturing Company. Inventory balançe items for all manufactured products are to be maintained on magnetic tape and represent one of the input files to this run. Since prices are not to be carried in this file, it is necessary at intervals during the year to apply current prices contained in a Price File which is the second input to the run. One output of run 4 is shown to be a Priced Inventory file containing all inventory items for which prices were found in the Price file. Since the two input files were maintained separately, it is possible that the Price file is not complete. Thus a second output file, the Unpriced Inventory, will contain those Inventory items for which no price was found. The system further requires that such unpriced items be printed on the Highspeed Printer so that they may be checked and the proper action taken.

The diagrams below show an actual example of the contents of each type of item in the problem.

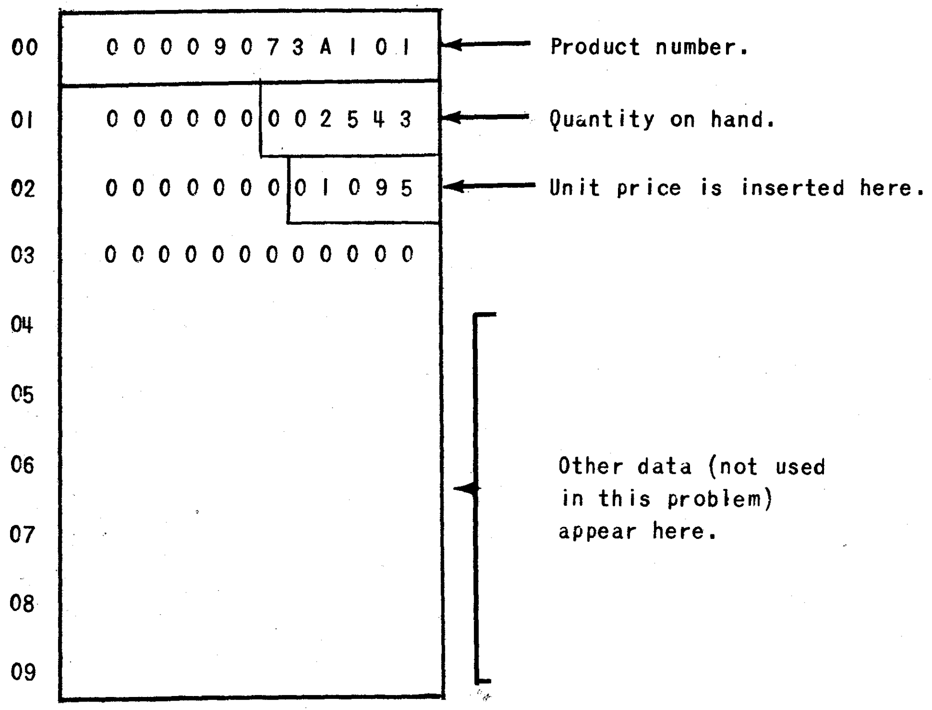
INVENTORY ITEM


00
0 0 0 0 9 0 7 3 A 1 0 1
This is the product number; it may require as many as 12 digits.
01
0 0 0 0 0 0 0 0 2 5 4 3
This field is the quantity of this kind of product on hand; it may require as many as 6 digits.
02
0 0 0 0 0 0 0 0 0 0 0 0
03
0 0 0 0 0 0 0 0 0 0 0 0
04
05
06
Other data (not used in this problem) appear here.
07
08
09

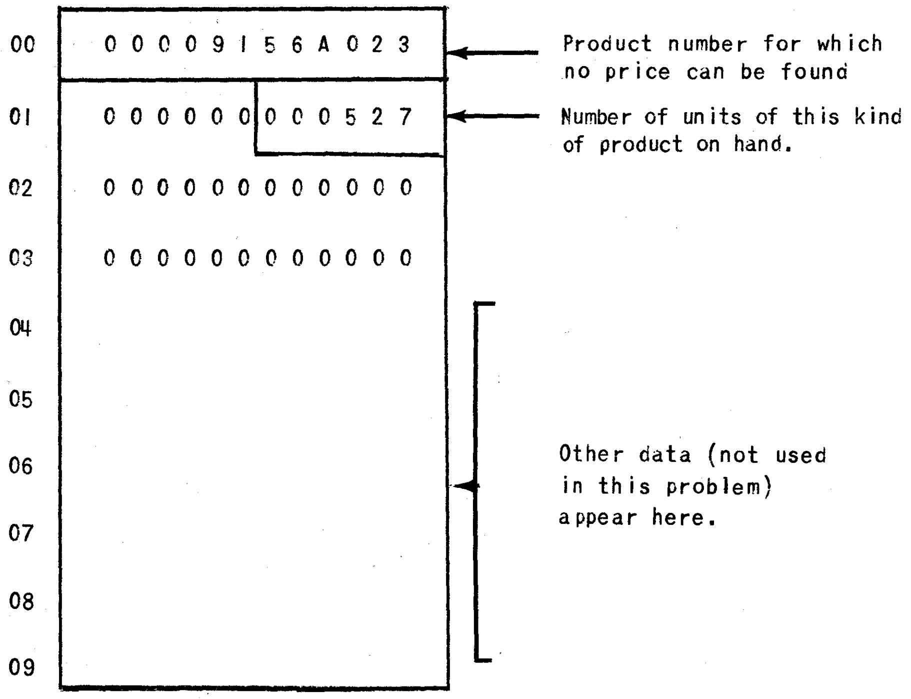
PRICE ITEM


00
0 0 0 0 9 0 7 3 A 1 0 1
This product number shows that this item applies to the inventory item given above.
01
0 0 0 0 0 0 0 0 1 0 9 5
This is the price for one unit of this product. Up to five digits may be used, and since this number represents dollars and cents, the decimal point is between the third and fourth digits - $10.95.

PRICED INVENTORY ITEM


00 0 0 0 0 9 0 7 3 A I 0 I
01 0 0 0 0 0 0 0 0 2 5 4 3
02 0 0 0 0 0 0 0 0 I 0 9 5
03 0 0 0 0 0 0 0 0 0 0 0 0
04
05
06
07
08
09
Product number.
Quantity on hand.
Unit price is inserted here.
Other data (not used in this problem) appear here.

UNPRICED INVENTORY ITEM


00 0 0 0 0 9 I 5 6 A 0 2 3
01 0 0 0 0 0 0 0 0 0 5 2 7
02 0 0 0 0 0 0 0 0 0 0 0 0
03 0 0 0 0 0 0 0 0 0 0 0 0
04
05
06
07
08
09
Product number for which no price can be found
Number of units of this kind of product on hand.
Other data (not used in this problem) appear here.

The key by which the price is matched with the corresponding inventory item is the product number. These files are sorted in ascending sequence according to product number by previous runs. The sorting runs are performed in order that the process of matching may be more efficiently carried out by UNIVAC in run 4.

The next step is to translate the description of the function of the run into a program for UNIVAC to follow in producing the desired output. It is in the logical analysis and organization of this program that the skilled programmer or systems analyst can make the most effective contribution. For it is in the overall logical design of the program that the application of programming and data processing know-how can make the difference between a correct program, and one which is both correct and efficient, in terms of making the best use both of the computer and of the data characteristics. ***FLOW-MATIC*** does not replace good systems design and careful run analysis.

A verbal outline of one possible program to solve this simple problem is given below.

Assume at the start that the first item of each of the input files is available. The operations are as follows:

I Compare the product number of the Inventory item with the product number of the Price item.

a. If the Inventory product number is less, go on to step II.
b. If the product numbers are equal, go on to step III.
c. If the Price product number is less, go on to step V.

II Prepare and write an Unpriced Inventory item. Then go on to step IV.

III Prepare and write a Priced Inventory item. Then go on to step IV.

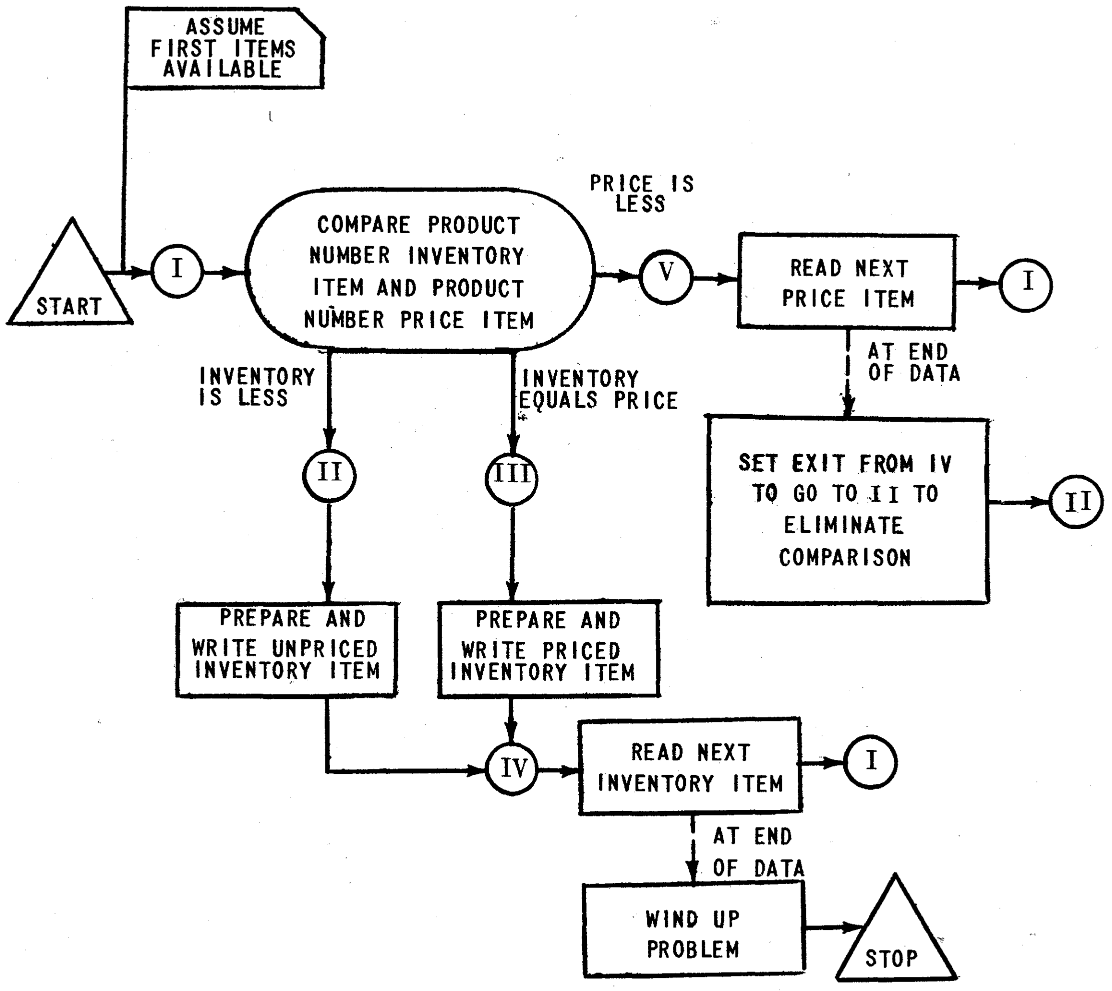
IV Read the next Inventory item and go back to step I. Or, if the Inventory file is exhausted, wind up the problem and stop.

V Read the next Price item and go back to step I. Or, if the Price File is exhausted, change the program so that step IV goes back to step II (eliminating a now unnecessary comparison). Then go on to step II.

In the preceding statements the functions of this simple run have been completely described. The next step is to translate these statements into a logical block chart for UNIVAC. (See Figure 16.)

BLOCK CHART

ABC MANUFACTURING COMPANY INVENTORY



ASSUME FIRST ITEMS AVAILABLE
START
I
COMPARE PRODUCT NUMBER INVENTORY ITEM AND PRODUCT NUMBER PRICE ITEM
PRICE IS LESS
V
READ NEXT PRICE ITEM
I
AT END OF DATA
SET EXIT FROM IV TO GO TO II TO ELIMINATE COMPARISON
II
INVENTORY IS LESS
INVENTORY EQUALS PRICE
II
III
PREPARE AND WRITE UNPRICED INVENTORY ITEM
PREPARE AND WRITE PRICED INVENTORY ITEM
IV
READ NEXT INVENTORY ITEM
I
AT END OF DATA
WIND UP PROBLEM
STOP

FIGURE 16

BLOCK CHART - The block chart is a diagram of the logical statements listed above. Such a diagram shows all of the paths which are required in a problem, and all intersections of the paths. The block chart should be checked for errors in logic, testing it with all variations of the data which are to be processed, and producing some sample output. Logical checking is done at this time because, while this version of the problem is concise, it must be complete and correct. The block chart, then, is by definition a complete, concise, correct diagram of the computer data processing procedure.

It is important to note that *FLOW-MATIC* does not replace the need for complete understanding of the job to be done. Nor does it replace the need for the careful analysis which must preceed the choice of the best computer procedure to do the job. The *FLOW-MATIC* system *does* make an indirect contribution to system design by facilitating the succeeding steps of flow-charting, coding, debugging (checking) and reprogramming, to such a degree that a considerably larger proportion of the total time and emphasis can be placed on run analysis and basic system design. By permitting the analyst or methods engineer to spend more time in block charting, *FLOW-MATIC* encourages the exercise of analytical ability and systems engineering experience.

Figure 16 is a block chart for run 4 which indicates a UNIVAC procedure to carry out the functions as specified by the procedural statements. Note the use of the names of functions, items, and data fields in the block chart which relate back to the original verbal description of the run. *FLOW-MATIC* encourages the use of words throughout the process of translation from descriptive English to machine language in order that the procedure which the UNIVAC Data Automation System carries out may be understandable to all involved in the development and use of the system. Although the block chart contains computer know-how, since it represents the application of computer characteristics and capacities to the requirements of the job, it is intelligible to the user and serves as a useful form of communication between the analyst and management.

The next step is one of determining the sequence of *FLOW-MATIC* operations equivalent to each block on the chart. A *FLOW-MATIC* operation is an operational unit designed to have maximum usefulness in data processing applications. For ease of use and recognition, these units are identified by English words and phrases. Insofar as possible, words have been chosen which are associated with the operations, in normal English language usage. For example, such terms as TEST, TRANSFER, JUMP, STOP are used. By maintaining English wording throughout the transition from procedural statement and definition to the complete *FLOW-MATIC* coded solution, the programmer and methods analyst are able to work directly in the terminology of the operations and procedure.

FLOW-MATIC CHART - The process of reducing the block chart to *FLOW-MATIC* sentences is most readily accomplished by drawing up a slightly expanded diagram, known as a *FLOW-MATIC* chart. This chart contains all of the *FLOW-MATIC* operations required to solve the problem, in their proper sequence.

To illustrate this process, consider one path, or branch, of the block chart, shown in Figure 17.

SAMPLE OF BLOCK CHART

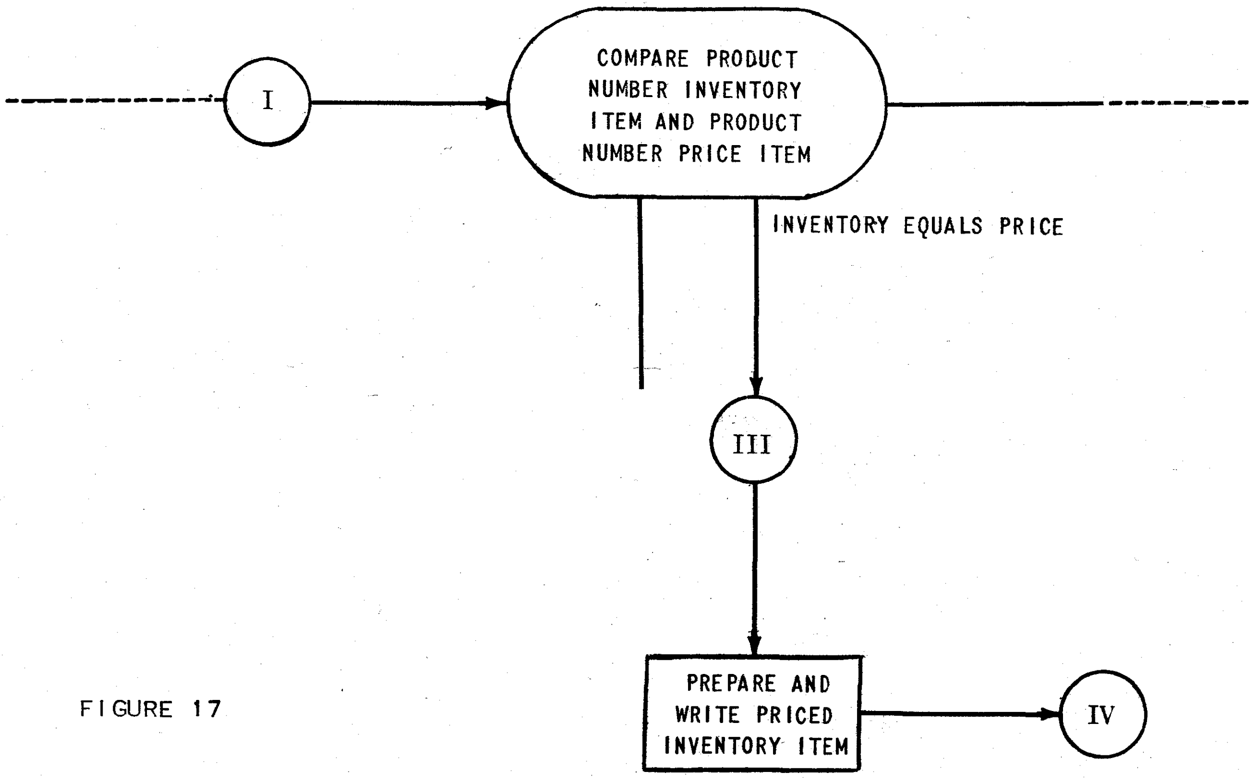

I
COMPARE PRODUCT NUMBER INVENTORY ITEM AND PRODUCT NUMBER PRICE ITEM
INVENTORY EQUALS PRICE
III
PREPARE AND WRITE PRICED INVENTORY ITEM
IV

FIGURE 17

Reference to the list of ***FLOW-MATIC*** operations in Appendix A, page 92 indicates that this path of the block chart may be expressed in ***FLOW-MATIC*** terms as shown in Figure 18. The ***FLOW-MATIC*** chart is completed in this general manner. by linking available ***FLOW-MATIC*** commands in the sequence required by the block chart.

SAMPLE OF FLOW-MATIC CHART

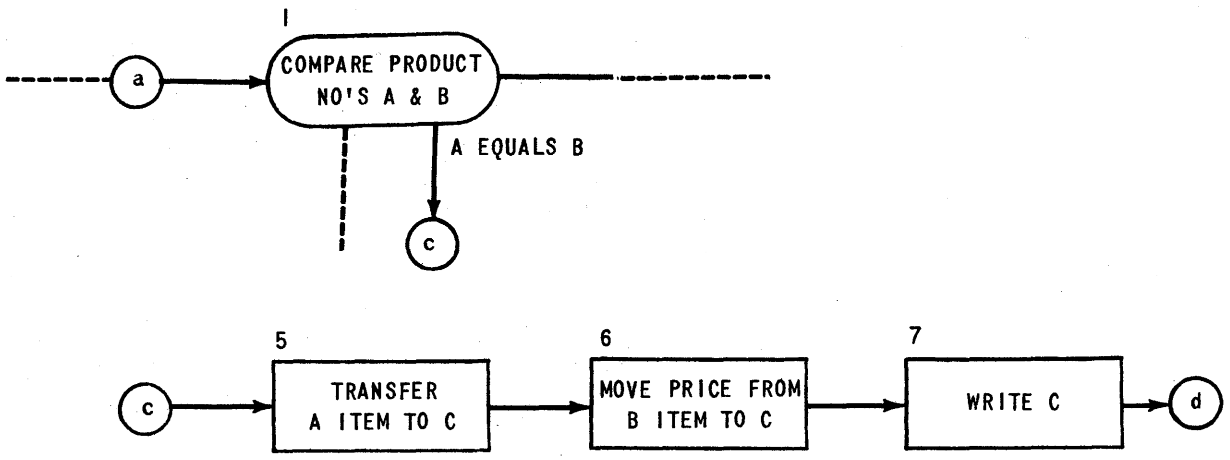


FIGURE 18

FLOW-MATIC CODE - Writing the ***FLOW-MATIC*** code for the problem consists of transcribing the ***FLOW-MATIC*** chart into a series of imperative sentences. For every operation, one statement is written, the format of which is given in Appendix A, pages 95-99. A sample of the code in its final form is below in Figure 19.

```
(1) COMPARE PRODUCT - NO (A) WITH PRODUCT - NO (B); IF GREATER GO TO OPERATION
    10; IF EQUAL GO TO OPERATION 5; OTHERWISE GO TO OPERATION 2 .

 .
 .
 .

(5) TRANSFER A TO C .

(6) MOVE UNIT-PRICE (B) TO UNIT-PRICE (C) .

(7) WRITE-ITEM C .
```

SAMPLE OF FLOW-MATIC CODE

FIGURE 19

DATA DESIGN: In order to produce a UNIVAC program, it is necessary that the compiler have information about the design of the input and output data files. The programmer supplies this information by filling out standard Data Design forms, shown in Appendix B. The Data Design information is recorded only once, even though the data files may be processed in a number of runs.

Information about the data files is divided into three categories; File, Item, and Field.

The File category contains information about the organization of the reel(s) of tape in the file; for example, whether or not the file is multireel, what the sentinel conventions are, and how the reel(s) are labelled. The item category lists the size of the item, and, if appropriate, states by what keys these items are sequenced. The field category provides a complete description of each field in the item; where it is located, how many digits it contains, and other related information.

The preprinted forms show how this information is to be listed, and, by providing the appropriate information the data is described for *FLOW-MATIC*. Data Designs are prepared for each of the two input files and the two output files in the Sample Problem run. By reference to the process chart (Figure 15, page 15) it can be seen that the input files, since they come from prior UNIVAC runs, have been previously defined. The *FLOW-MATIC* System provides for storage of such Data Designs on tape so that they may be called upon for reuse.

COMPILATION - When the *FLOW-MATIC* code and the Data Designs have been written, the problem is ready for compilation. It is only necessary to transcribe onto tape the information which has been prepared. The layout of the tape prepared on a Unityper is shown in Figure 20, where it is displayed as the input tape to the *FLOW-MATIC* compilation.

The *FLOW-MATIC* compiler delivers, as output, a complete UNIVAC program tape, which can be immediately tested by mounting it, together with the appropriate data tapes, and making a trial run. Since the coding which controls the movement of the files through the computer is provided by *FLOW-MATIC*, and since each section of machine code generated by *FLOW-MATIC* is correct by itself, the program will run if errors in logical analysis or errors undetected in proof-reading are not present. Even *FLOW-MATIC* cannot protect the analyst from mistakes in logical analysis. Many internal inconsistencies in the *FLOW-MATIC* input code will be detected by the *FLOW-MATIC* System during the conversion to computer coding. Logical errors can be located by reference to the *FLOW-MATIC* chart. For locating those program errors not discovered by any of the previously cited means, the compiler provides an Edited Record of compilation, which enables the analyst with an assist from a programmer, to relate the computer coding for the problem back to the *FLOW-MATIC* coded statements or to the chart.

FLOW-MATIC COMPILER RUN CHART

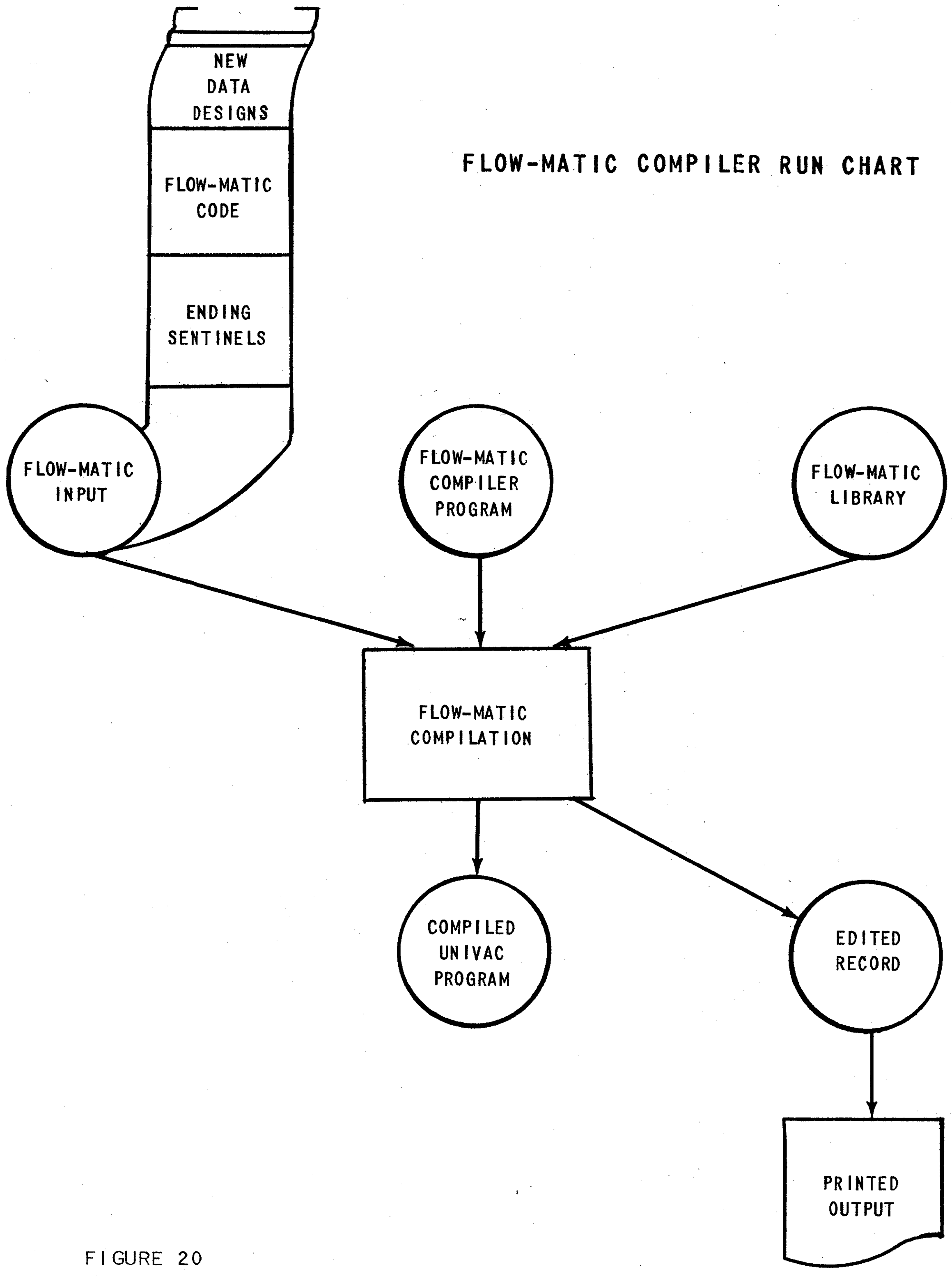


FIGURE 20

Chapters 3 and 4 trace, in detail, the preparation of the *FLOW-MATIC* code and Data Designs for the Sample Problem. Chapters 5 and 6 introduce problem variations which show additional facilities of the *FLOW-MATIC* compiler. Chapter 7 describes the production and testing of the compiled program.

chapter 3

Flow-Matic Charting And Program-Writing

The complete *FLOW-MATIC* chart for Sample Problem 1 described in Chapter 2 is created from the process chart and the block chart with the aid of the list of *FLOW-MATIC* code operations given in Appendix A, page 92. Writing the *FLOW-MATIC* chart is, in fact, simply a matter of choosing the necessary functions from the list on Appendix A, page 92, and arranging them in the order prescribed in the block chart. The *FLOW-MATIC* chart is used primarily as a guide for writing the *FLOW-MATIC* code, and it is simply a restatement of the logic of the problem in terms easily adaptable to the code.

Figure 21 shows a partially completed *FLOW-MATIC* chart. All of the required functions are indicated, and their sequence is shown. Several useful conventions are employed. Circles, here labelled by lettering, indicate interconnections between paths; flags are used to assert a condition which exists at a given point; dotted lines indicate points at which a conditional change in sequence can occur; rectangular boxes are used for evaluation of a formula or straight computation and ovals are for deciding among one of various paths of computational flow, based upon the equality and/or magnitude of two quantities. Note that comments on this chart are still largely in English, and that no numbers have yet been used.

INITIAL VERSION OF FLOW-MATIC CHART

ABC MANUFACTURING COMPANY INVENTORY - SAMPLE PROBLEM I

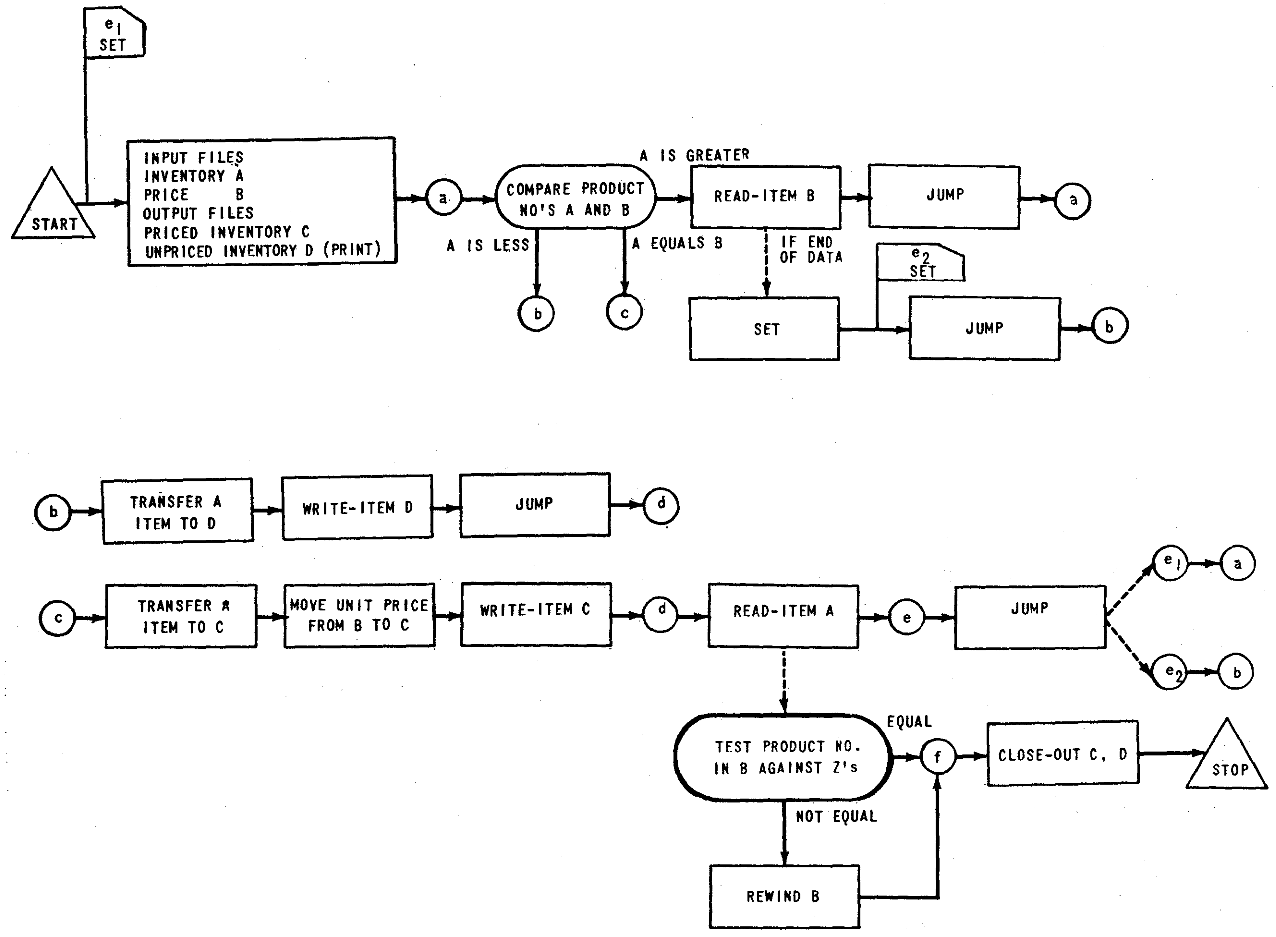


FIGURE 21

The logical order of performance of these operations is now indicated by attaching a number to each of the chosen functions. There are four general rules pertaining to these numbers and their sequence:

1. The operation number sequence starts with zero, and operation zero always specifies the input and output files to be processed by the program.

2. The operation numbers are assigned in unbroken sequence.

3. The operations are to be performed in numerical sequence, unless a specific statement to the contrary is made.

4. The highest number is assigned to the operation which stops the problem.

Observing these rules, the *FLOW-MATIC* chart is completed by numbering the boxes for each operation, and inserting the correct operation numbers at branch points.

Figure 22 shows the completed *FLOW-MATIC* chart for the Sample Problem. There are several features to notice in examining this chart.

The files in the problem are labelled with the letters A, B, C, and D. This is done for ease of reference and finds further use in writing the *FLOW-MATIC* code.

The action to be taken when each of the input files is exhausted is directly connected to the operation of obtaining a new item. This is done because *any* request for a new item from a file may exhaust the data.

The test indicated in operation fourteen determines whether or not the Price File has been exhausted when the end of the Inventory data has been reached. A sentinel, a symbol made up of some combination of characters which cannot appear as valid data, marking the end of the file is placed at the end of each file to indicate the end of the data. The test therefore determines whether or not this sentinel is present in the current Price File item.

Operation twelve shows a logical operation to be performed upon the program rather than on the data being processed. It directs that hereafter operation nine will return not to operation one, but to operation two, bypassing a now unnecessary step and directing all further Inventory items to the Unpriced Inventory File.

FINAL VERSION OF FLOW-MATIC CHART

ABC MANUFACTURING COMPANY INVENTORY - SAMPLE PROBLEM I

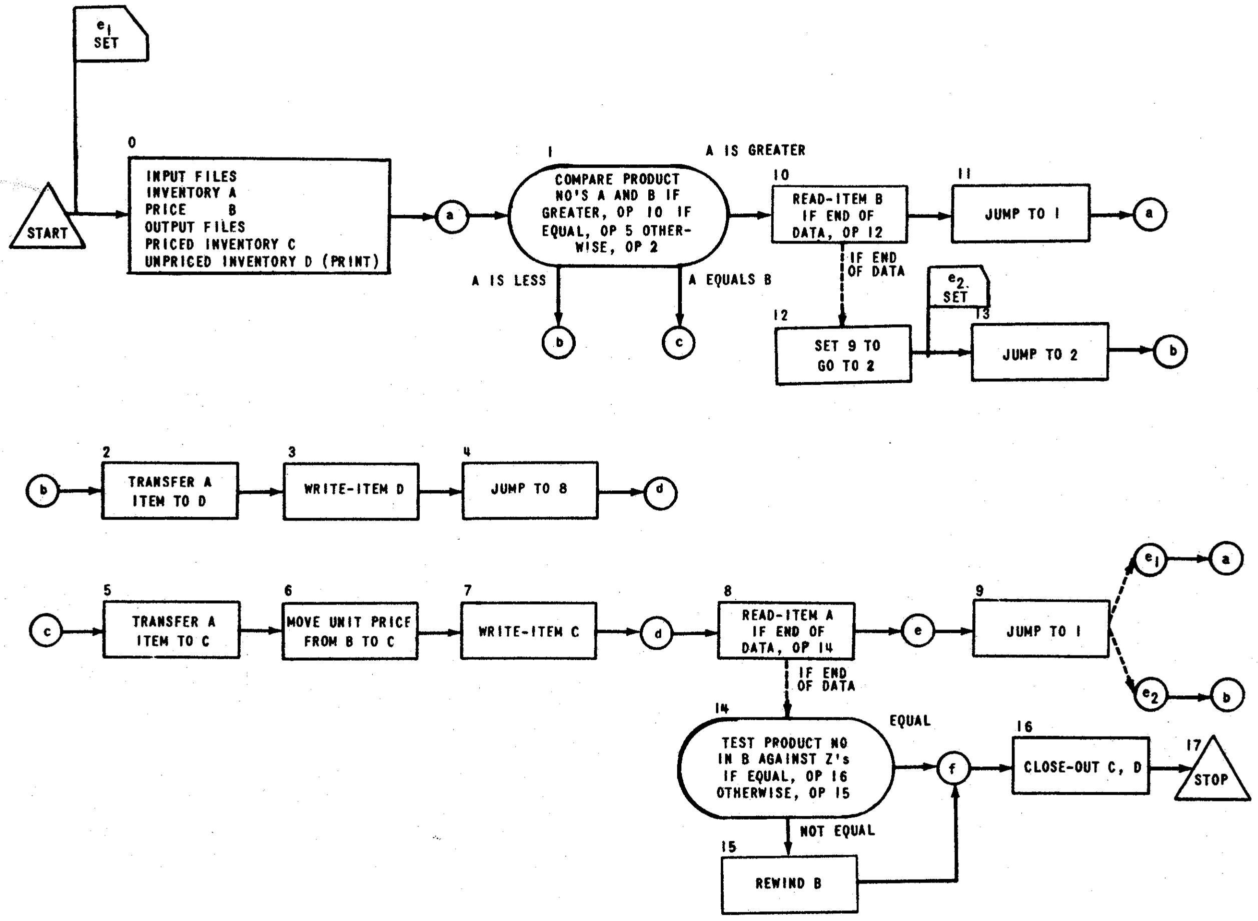


FIGURE 22

This completed chart (Figure 22 on page 29) is directly convertible to *FLOW-MATIC* coding. Writing the sequence of English sentences which are the code for the problem is done by cross-referencing between the chart and the description of the available statements.

These statements follow common English usage in punctuation and format, that is, words are separated by spaces, and a period is used to terminate each sentence. In addition, each statement is labelled with its operation number as given in the chart.

The programmer modifies the format of each statement to adapt it to his problem by choosing the correct option for his purpose, and assigning his own names to the fields and files operated upon.

The general rules which apply to *FLOW-MATIC* coding are listed below:

(1) A file name assigned by the programmer may be a maximum of twelve digits in length, and none of these digits may be a space. If it is desirable to combine English words into one name, hyphens may be used, e. g., (0) INPUT INVENTORY FILE-A PRICE FILE-B;.......
...PRICED-INV FILE-C...

(2) An item name is simply the assigned letter of its file, e.g., (10) READ-ITEM B...

(3) A field name is similar to a file name, twelve or fewer non-space digits. But in addition, each field name is modified by its file letter in parentheses, e.g., (1) COMPARE PRODUCT-NO (A) WITH......

(4) Operation numbers labelling *FLOW-MATIC* statements are parenthesized. Those appearing in the body of the statement are not parenthesized, e.g., (9) JUMP TO OPERATION 1 .

In addition to the list of functions, Appendix A gives the precise format for each function, and a summary of the rules applying to the use of these statements. In reading the following descriptions of the *FLOW-MATIC* statements required for the Sample Problem, reference to Appendix A will clarify the manner in which they are written. In each of the statements shown below, the information supplied by the programmer is underlined.

```
(0) INPUT INVENTORY FILE-A PRICE FILE-B ;
    OUTPUT PRICED-INV FILE-C UNPRICED-INV FILE-D ;
    HSP D .
```

Remembering that operation zero specifies the input and output files for the problem, the function INPUT is used. Choosing the format for two input files and two output files, the programmer assigns names and letters to these files (shown underlined in the above statement). He also specifies that the UNPRICED-INV file labelled D is to be printed on the UNIVAC High Speed Printer.

Note that the code for operation zero is a description of the process chart (Figure 15, page 15) for the run.

This statement has two major functions:

(a) To start the movement of the specified data files through the computer in their proper sequence, and

(b) To label the data files with letters so that hereafter these files, and their fields and items, may be referred to by the assigned letter.

It should be noted that this version of the initial statement is only one of many possibilities. It will vary as the process chart varies from one run to another. For example, the number of inputs and outputs is variable, the specific Uniservos to be used may be given, or it may be stated that a file is to be prepared for conversion to punched cards, and so on.

(1) COMPARE PRODUCT-NO (A) WITH PRODUCT-NO (B) ; IF GREATER GO TO OPERATION 10 ; IF EQUAL GO TO OPERATION 5 ; OTHERWISE GO TO OPERATION 2 .

Since the logical operation numbered one in the *FLOW-MATIC* chart calls for comparison of two fields, the programmer uses the order COMPARE. The three conditions required are stated by using the fourth option listed in the description of this operation. Having chosen the correct format, the programmer inserts the assigned names for the fields, and indicates, by the assigned letter, to which file he is referring. Note that the convention of labelling fields with their file letters allows the programmer to use the same name for fields from different files. Within one file, however, each field is given a unique name.

It remains to specify which operation is to be performed next in each of the three cases which occur. This is done by direct reference to the *FLOW-MATIC CHART*.

(2) TRANSFER A TO D .
Operation two in the chart calls for the transfer of a complete item from the file lettered A (INVENTORY) to the file lettered D (UNPRICED-INV). The TRANSFER order accomplishes this.

(3) WRITE-ITEM D .
Operation three indicates that the current item in the D file (UN-PRICED-INV) is to be recorded on magnetic tape.

(4) JUMP TO OPERATION 8 .
In operation four the chart indicates a break in the normal sequence. The programmer fills in from the chart the operation number which is to be performed next.

(5) TRANSFER A TO C .
The logical function to be performed is the same as in operation two, but here the A item (from the INVENTORY file) is to be moved to file C (PRICED-INV).

(6) MOVE UNIT-PRICE (B) TO UNIT-PRICE (C) .
The function of this operation is to insert the price from file B (the PRICE file) into the C item created by operation six. The programmer assigns field names (here the same name, UNIT-PRICE, is used in both files) and attaches the proper file letters.

(7) WRITE-ITEM C .
In this operation the completed item in the C file is to be recorded on magnetic tape. See operation three.

(8) READ-ITEM A ; IF END OF DATA GO TO OPERATION 14 .
The function of this statement is to obtain the next consecutive item from file A (INVENTORY). Since a request for the next item may exhaust the data file, the programmer states as a part of this function which operation is to be performed when the end of the data is encountered. In this case, the chart shows that 14 is the number required.

(9) JUMP TO OPERATION 1 .
Operation nine in the *FLOW-MATIC* chart indicates a break in the normal sequence, that is, that operation number 1 is to be performed next.

(10) READ-ITEM B ; IF END OF DATA GO TO OPERATION 12 .
This operation is to obtain the next consecutive item from file B and also to indicate that operation number 12 is to be performed when the end of the file is encountered. See also operation eight.

(11) JUMP TO OPERATION 1 .
See operation four.

(12) SET OPERATION 9 TO GO TO OPERATION 2 .
The ***FLOW-MATIC*** chart indicates that the function here is to alter another operation in the sequence, namely number nine. Operation nine as originally stated is a jump to number 1. The current operation (twelve) is to change nine so that it becomes a jump to operation two.

(13) JUMP TO OPERATION 2 .
See operation four.

(14) TEST PRODUCT-NO (B) AGAINST ZZZZZZZZZZZZ ; IF EQUAL GO TO OPERATION 16 ; OTHERWISE GO TO OPERATION 15 .

Operation fourteen in the ***FLOW-MATIC*** chart calls for a comparison of a data field with a constant quantity, a word of Zs. This is a different logical function from the comparison of two data fields, as was done in operation one. Here the required order is TEST rather than COMPARE. The programmer chooses the option whose format provides the conditions indicated by the chart. He then inserts the name of the field to be tested, PRODUCT-NO, with its file letter, B. The constant quantity is indicated by inserting its actual value. From the chart the programmer determines the operation number to be performed in the two cases which occur.

(15) REWIND B .
Operation fifteen states that the current reel of the file lettered B is to be rewound. Here the logic of the problem is such that, although the end of the B data has not been encountered, the last applicable item has been used and the file can be terminated.

(16) CLOSE-OUT FILES C , D .
The CLOSE-OUT order of operation sixteen calls for the termination of the two output files C and D.

(17) STOP . (END)
The last operation in the sequence is the STOP order, and it is always followed by the word END in parentheses.

Figure 23 shows the completed ***FLOW-MATIC*** code, as it is submitted for Unitypping. The unused portion of the block is space-filled.

FLOW-MATIC CODE
FOR SAMPLE PROBLEM I

```
(0)   INPUT INVENTORY FILE-A PRICE FILE-B ; OUTPUT PRICED-INV FILE-C UNPRICED-
      INV FILE-D ; HSP D .
(1)   COMPARE PRODUCT-NO (A) WITH PRODUCT-NO (B) ; IF GREATER GO TO OPERATION
      10 ; IF EQUAL GO TO OPERATION 5 ; OTHERWISE GO TO OPERATION 2 .
(2)   TRANSFER A TO D .
(3)   WRITE-ITEM D .
(4)   JUMP TO OPERATION 8 .
(5)   TRANSFER A TO C .
(6)   MOVE UNIT-PRICE (B) TO UNIT-PRICE (C) .
(7)   WRITE-ITEM C .
(8)   READ-ITEM A ; IF END OF DATA GO TO OPERATION 14 .
(9)   JUMP TO OPERATION 1 .
(10)  READ-ITEM B ; IF END OF DATA GO TO OPERATION 12 .
(11)  JUMP TO OPERATION 1 .
(12)  SET OPERATION 9 TO GO TO OPERATION 2 .
(13)  JUMP TO OPERATION 2 .
(14)  TEST PRODUCT-NO (B) AGAINST ZZZZZZZZZZZZ ; IF EQUAL GO TO OPERATION 16 ;
      OTHERWISE GO TO OPERATION 15 .
(15)  REWIND B .
(16)  CLOSE-OUT FILES C , D .
(17)  STOP . (END) Space Fill to End of Block.
```

ABC MANUFACTURING COMPANY INVENTORY

FIGURE 23

chapter 4

File-Data Layout And Design

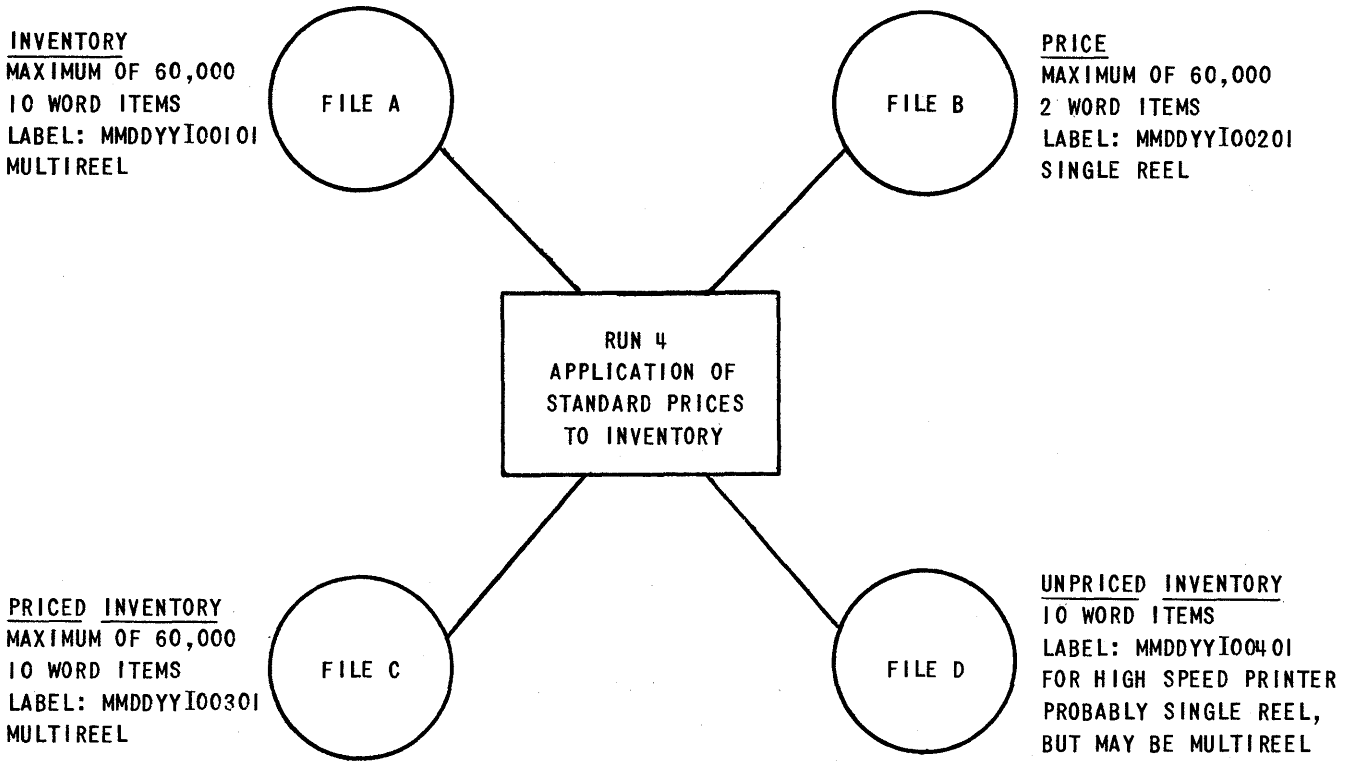
The run description in Chapter 2 included a complete statement of the logical function to be performed. The *FLOW-MATIC* charting and coding can be completed from this statement, but before the Data Designs can be completed, several detailed questions must be answered.

Figure 24 is a copy of the run chart presented in Chapter 2 with additional details shown (e.g., item sizes, labels, whether single reel or not). Figure 25 illustrates the item layouts for the data files involved in this problem. It should be noted that the use of *FLOW-MATIC* permits postponement of the process of making item layouts until the *FLOW-MATIC* Code is written. The system does not place any limits on item layouts, but it does not eliminate the need for them.

With this information available, Data Designs can be prepared on the preprinted forms illustrated (Appendix B). Separate forms for the File, Item, and Field designs are provided at the end of this chapter. These preprinted forms allow for future expansion of the *FLOW-MATIC* System.

Provision has been made in the *FLOW-MATIC* System for storing Data Design information about specific files on the *FLOW-MATIC* library tape. Thus if a system of related runs is to be programmed using *FLOW-MATIC*, Data Designs for files which will be processed in more than one run can be stored and called upon any number of times in *FLOW-MATIC* programs. Using this facility, there is no necessity to fill out Data Design information for files already described. Since input files to one run commonly are output files of other runs, a significant reduction of (clerical) effort in writing and recording these Data Designs is possible.

PROCESS CHART
ABC MANUFACTURING COMPANY INVENTORY
ALL FILES SEQUENCED BY PRODUCT NUMBER



CONVENTIONS

(1) LABELS IN WORD 03 OF FIRST BLOCK ON EACH REEL.

(2) BLOCK COUNTS IN WORD 01 OF LAST ITEM IN SENTINEL BLOCK.

(3) SENTINELS ARE ZZZZZZZZZZZY FOR END OF REEL AND ZZZZZZZZZZZZ FOR END OF FILE. THESE ARE LOCATED IN THE KEY WORD POSITION (WORD 000) OF FIRST INVALID ITEM AND LAST ITEM OF SENTINEL BLOCK.

FIGURE 24

ABC MANUFACTURING COMPANY INVENTORY
ITEM LAYOUTS FOR RUN 4

FILE A (INVENTORY)

```
00   P P P P P P P P P P P P
01   0 0 0 0 0 0 Q Q Q Q Q Q
02   0 0 0 0 0 0 0 0 0 0 0 0
03   0 0 0 0 0 0 0 0 0 0 0 0
04  [                        ]
05  [                        ]
06  [      OTHER DATA        ]
07  [                        ]
08  [                        ]
09  [                        ]
```

FILE B (PRICE)

```
00   P P P P P P P P P P P P
01   0 0 0 0 0 0 0 U U U U U
                        ^
```

P = PRODUCT NUMBER

U = UNIT PRICE

Q = QUANTITY ON HAND

E = EXTENDED PRICE
(SEE CHAPTER 6)

^ = LOCATION OF DECIMAL POINT

FILE C (PRICED INVENTORY)

```
00   P P P P P P P P P P P P
01   0 0 0 0 0 0 Q Q Q Q Q Q
02   0 0 0 0 0 0 0 U U U U U
                        ^
03   0 0 E E E E E E E E E E
                        ^
04  [                        ]
05  [                        ]
06  [      OTHER DATA        ]
07  [                        ]
08  [                        ]
09  [                        ]
```

FILE D (UNPRICED INVENTORY)

```
00   P P P P P P P P P P P P
01   0 0 0 0 0 0 Q Q Q Q Q Q
02   0 0 0 0 0 0 0 0 0 0 0 0
03   0 0 0 0 0 0 0 0 0 0 0 0
04  [                        ]
05  [                        ]
06  [      OTHER DATA        ]
07  [                        ]
08  [                        ]
09  [                        ]
```

FIGURE 25

NAME OF FILE

The programmer writes in the file name which may not start with the word FILE.

FILE DESIGN*

This section of the Data Designs contains information about the organization of the file. The data file conventions to be used are listed here and they determine the details of the coding which will be produced to control the file.

Nine two-word packets are required. Additional two-word entries may be made if it becomes necessary to describe other features of the file. Each two-word packet is listed and described below.

The first packet is:

L	A	B	E	L	Δ	Δ	Δ	Δ	Δ	Δ	Δ
L	L	L	L	L	L	L	L	L	L	L	L

LLLLLLLLLLLL represents the identification that has been assigned to this data file. The least significant digits should end in the digit 1, indicating reel one. Allowance may be made for numbering as many reels as are required. (If the maximum number of reels in a file is 9, one digit is allocated as a counter; 99, two digits; 999, three digits; etc.) If the label is not known or if it is not desired to specify the label, twelve spaces are used.

The second packet is:

L	O	C	Δ	O	F	Δ	L	A	B	E	L
0	0	0	0	0	0	0	0	0	W	W	W

WWW indicates the word position of the label within the label block. The sixty words in the label block are addressed as 000-059.

* Consult figures on pages 46, 49, 52 and 55.

The third packet is:

M	U	L	T	I	Δ	R	E	E	L	Δ	Δ
0	0	0	0	0	0	0	0	0	0	0	n

The digit n may be either zero or one. If zero, it means that this file will not exceed one reel. A one is used in all cases where files may exceed one reel.

The fourth packet is:

B	L	K	Δ	C	T	Δ	I	N	D	Δ	Δ
0	0	0	0	0	0	0	0	0	0	0	b

b may be either zero or one. If one, it means that the tally of the number of blocks is maintained (for input reels) or will be written (on output reels) on the end of each reel of this file. On input reels the block count recorded on the reel is checked against a counter maintained during the processing. *All* blocks on the data tape will be counted, including label block, data blocks and sentinel blocks. Use b = zero to indicate that this control feature is not desired.

The fifth packet is:

B	L	K	Δ	C	T	Δ	L	O	C	Δ	Δ
0	0	0	0	0	0	0	0	0	X	X	X

XXX indicates the word position of the block count *within the last item* of the sentinel block. The allowable range for XXX is from ooo through item-size minus one. If the block count indicator is zero, this entry is not significant and the word is filled with zeros.

The sixth packet is:

E	N	D	Δ	R	E	E	L	Δ	S	E	N
S	S	S	S	S	S	S	S	S	S	S	S

SSSSSSSSSSSS represents the twelve digits used to indicate end of valid data on intermediate reels, e.g., ZZZZZZZZZZZY . If the multi-reel indicator is zero, spaces are written here.

The seventh packet is:

E	N	D	Δ	F	I	L	E	Δ	S	E	N
S	S	S	S	S	S	S	S	S	S	S	S

SSSSSSSSSSSS represents the twelve digits used to indicate end of valid data on the *final* reel of a file, e.g., ZZZZZZZZZZZZ

The eighth packet is:

L	O	C	Δ	I	N	Δ	F	I	R	S	T
0	0	0	0	0	0	0	0	0	W	W	W

WWW indicates the word position of the end sentinels within the first invalid item of the sentinel block. The allowable range for WWW is from ooo through item size minus one.

The ninth packet is:

L	O	C	Δ	I	N	Δ	L	A	S	T	Δ
0	0	0	0	0	0	0	0	0	W	W	W

WWW indicates the word position of the end sentinels within the last item of the sentinel block. The allowable range for WWW is from ooo through item size minus one.

Although *FLOW-MATIC* permits flexibility in the conventions to be applied to tape files, it is expected that in a given installation much of this information will remain fixed, thereby providing automatic standardization. The flexibility is still necessary when the files are to be processed by runs other than those produced by *FLOW-MATIC*, e.g., sorts.

ITEM DESIGN*

This section of the DATA DESIGNS contains information about the data items in the file.

Three two-word packets of information are required, and more can be added on an optional basis.

The first packet is:

I	T	E	M	Δ	S	I	Z	E	Δ	Δ	Δ
0	0	0	0	0	0	0	0	0	n	n	n

nnn represents the size of the data items in the file, e.g., the item in this sample problem is a ten word item (see page 47).

The second packet is:

N	O̲	Δ	O̲	F	Δ	K	E	Y	S	Δ	Δ
0	0	0	0	0	0	0	0	0	0	0	k

k is a digit from zero through nine. The key is the field or fields by which the file is sequenced. If a file is sequenced by a major key such as last name, and a minor key such as first name, k would equal 2.

The third packet is:

K	E	Y	Δ	1	Δ	Δ	Δ	Δ	Δ	Δ	Δ
Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y

YYYYYYYYYYYY represents the assigned name of the field which is the major key in this file. If the file is unsequenced, the k digit in the previous packet is zero and YYYYYYYYYYYY is twelve spaces.

When k is greater than one, additional packets are required for each additional key in the order of decreasing significance.

* Consult figures on pages 47, 50, 53 and 56.

If "k" in NO. OF KEYS were 2, then a fourth packet would be required, following the same format as the third packet:

```
KEY△2△△△△△△△
YYYYYYYYYYYY
```

Each field named as a key must be listed in the Field Design section under the same name used here.

In the preprinted forms in Appendix B additional information may be inserted following the name of the last key. In some problems, for example, it is desirable to treat a consecutive group of UNIVAC words within the item as a sub-item. Rather than calling for this sub-item by mentioning all of the names of the fields within it, a new name is applied to the sub-item itself. This assigned name is then entered in the Data Design for the file in the following way:

```
AAAAAAAAAAAA
000SSS000EEE
```

AAAAAAAAAAAA is the name chosen by the programmer, e.g., ADDRESS△△△△△ .
SSS is the relative word position of the first word of the sub-item within the item. The allowable range is 000 through item size minus one.
EEE is the position of the last word of the sub-item within the item. The allowable range is 000 through item size minus one.

As many such sub-items as are required by the problem may be so named following the name of the last key.

FIELD DESIGN*

This section of the Data Designs contains detailed information about the data fields within the item.

If the Data Design is to be used in many runs, all fields in the item should be described. If the Data Design is to be used to compile only one run, only those fields mentioned in the *FLOW-MATIC* code for the run need be described. Each field in the item is described with a four word packet.

YYYYYYYYYYYY	=	Name of field
000WWW000000	=	Word Location in item
00000TPPSLN0	=	Field Descriptor
EEEEEEEEEEEE	=	Extractor

YYYYYYYYYYYY in the first word represents the name of the field exactly as it appears in the *FLOW-MATIC* code. It may be a maximum of twelve consecutive non-space digits. If the name contains fewer than twelve digits, unused digits to the right are space filled. If names consist of two or more parts, these parts are separated by hyphens; e.g., PRODUCT-NO.

WWW in the second word represents the word position of this field within the item, i.e., for a ten word item the words are 000-009.

TPPSLN in the third word:

T in the 6th digit of the 3rd word represents the type of field.
T = 1 = alphabetic
T = 2 = alpha-numeric
T = 3 = numeric

PP in 3rd word indicates the location of the assumed decimal point relative to a reference point immediately to the left of the field. The position just to the left of the field is indicated by PP equal to 00.

If the assumed decimal is one position to the left of the reference point, PP is 1L. If the assumed decimal is one position to the right of the reference point, PP is 1R. Assumed desimal points may be positioned a maximum of 35 places to the left or to the right, (e.g., ƵL = 35 left
ML = 22 left
AL = 10 left
AR = 10 right
JR = 19 right
ƵR = 35 right).

If the field has no assumed decimal point, the digits PP are written as ignores. (*ii*)

* Consult figures on pages 48, 51, 54 and 57.

S in 3rd word represents the location of the sign of this number. Ignore (/) is used for fields without signs. A numeric field with the S digit equal to ignore is assumed to be a positive number.

If the field has a sign, the S digit may be 1 through 9, A, B, or C. The twelve digit positions within a UNIVAC word are labelled.

1	2	3	4	5	6	7	8	9	A	B	C

L in the 3rd word represents the location within the UNIVAC word of the left most digit of the field excluding its sign. As mentioned above, L may be 1 through 9, A, B, or C.

N in 3rd word represents the number of consecutive (adjacent) digits in the field, excluding its sign. N may equal 1 through 9, A, B, or C, where A equals 10, B equals 11, and C equals 12 digits.

EEEEEEEEEEEE in the fourth word is a pattern of ones and zeros, showing the digit locations occupied by the field, *including* the sign digit if any. Ones indicate the digits comprising the field, and all other digits are zeros.

If the field is alone in the word and can be treated as a whole word, twelve zeros are used, not twelve ones.

Although the fields described for the sample problem all appear in separate UNIVAC words in the item, the more common situation is that several fields are packed in one word. In *FLOW-MATIC*, each such field has its own name and description in the Data Designs.

It also may happen that two fields overlap each other within the UNIVAC word. For example, consider the field PRODUCT-NO in the Inventory File. The last three digits of this field may be a type number, not used in this problem as such. In a problem where the type number must be treated as well as the product number, both fields are described, as shown below:

P	P	P	P	P	P	P	P	P	P	P	P
									T	T	T

```
PRODUCT-NOΔΔ
000000000000        Four word packet describing 12
000002///1C0        digit field.
000000000000

TYPE-NUMBERΔ
000000000000        Four word packet describing 3
000003///A30        digit field within 12 digit field.
000000000111
```

Sentinels

Immediately following the ***last*** four word packet, ***and*** in word 59 of the last block of the Data Design, the sentinel

ENDΔFILEΔDES

is written. The rest of the block is filled with zeros.

After the ***FLOW-MATIC*** Code for the program has been written (Chapter 3), and the Data Designs have been filled in as described in this chapter and shown in Figures 26 a through 26 l, the input tape for ***FLOW-MATIC*** compilation may be UNITYPED. Figure 27 shows a HIGH-SPEED PRINTER copy of this information just as it appears, ready to compile the program for Sample Problem 1.

ANALYST: ____________ **INPUT AND OUTPUT DATA DESIGN** PAGE: ______

DATE: ____________ **ABC MANUFACTURING COMPANY INVENTORY PROBLEM**

N	A	M	E	Δ	O	F	Δ	F	I	L	E	
I	N	V	E	N	T	O	R	Y	Δ	Δ	Δ	English name of file.
F	I	L	E	Δ	D	E	S	I	G	N	Δ	
Δ	Δ	Δ	Δ	Δ	Δ	Δ	Δ	Δ	Δ	Δ	Δ	
L	A	B	E	L	Δ	Δ	Δ	Δ	Δ	Δ	Δ	
M	M	D	D	Y	Y	1	0	0	1	0	1	1.) Label with reel counter. 2.) If label variable all Δ's.
L	O	C	Δ	O	F	Δ	L	A	B	E	L	
0	0	0	0	0	0	0	0	0	0	0	3	Word location of label in label block.
M	U	L	T	I	Δ	R	E	E	L	Δ	Δ	
0	0	0	0	0	0	0	0	0	0	0	1	1 = Yes; 0 = No.
B	L	K	Δ	C	T	Δ	I	N	D	Δ	Δ	
0	0	0	0	0	0	0	0	0	0	0	1	1 = Block count desired; 0 = No block count.
B	L	K	Δ	C	T	Δ	L	O	C	Δ	Δ	
0	0	0	0	0	0	0	0	0	0	0	1	Word location of block count in last item of sentinel block. [000 to (item size-1)]
E	N	D	Δ	R	E	E	L	Δ	S	E	N	
Z	Z	Z	Z	Z	Z	Z	Z	Z	Z	Z	Y	End of reel sentinel. If single reel, all Δ's.
E	N	D	Δ	F	I	L	E	Δ	S	E	N	
Z	Z	Z	Z	Z	Z	Z	Z	Z	Z	Z	Z	End of file sentinel.
L	O	C	Δ	I	N	Δ	F	I	R	S	T	
0	0	0	0	0	0	0	0	0	0	0	0	Word location of sentinel in first invalid item. [000 to (item size-1)]
L	O	C	Δ	I	N	Δ	L	A	S	T	Δ	
0	0	0	0	0	0	0	0	0	0	0	0	Word location of sentinel in last item of sentinel block. [000 to (item size-1)]

Other entries may be added here, each consisting of a title word and an information word; e.g., PARTΔBLKΔSEN
ZZZZZZZZZZZX

UNITYPIST NOTE: After the last entry skip to the next page.

FLOW-MATIC DATA DESIGN FORM 1

Remington Rand Univac
DIVISION OF SPERRY RAND CORPORATION

FIGURE 26a

ANALYST: ______________________ PAGE: ____

DATE: ______________________

I	T	E	M	Δ	D	E	S	I	G	N	Δ	
Δ	Δ	Δ	Δ	Δ	Δ	Δ	Δ	Δ	Δ	Δ	Δ	
I	T	E	M	Δ	S	I	Z	E	Δ	Δ	Δ	
0	0	0	0	0	0	0	0	0	0	1	0	1,2,3,4,5,6,10,12,15,20,30,60.
N	O	Δ	O	F	Δ	K	E	Y	S	Δ	Δ	
0	0	0	0	0	0	0	0	0	0	0	1	0,1,2,....,9 = number of keys
K	E	Y	Δ	1	Δ	Δ	Δ	Δ	Δ	Δ	Δ	
P	R	O	D	U	C	T	-	N	O	Δ	Δ	Name of field, if no key ΔΔΔΔΔΔΔΔΔΔΔΔ

Further key entries may be added here, each consisting of KEYΔnΔΔΔΔΔΔΔ followed by the name of the field.

If there are Sub-items to be described, the descriptions are entered following the last Key entry. Sub-items are described with two-word packets consisting of the name of the Sub-item followed by a word in the format 000SSS000EEE, where SSS is the first word and EEE is the last word of the Sub-item, relative to the entire item.

UNITYPIST
NOTE: After the last entry skip to the next page.

FLOW-MATIC DATA DESIGN FORM 2

Remington Rand Univac
DIVISION OF SPERRY RAND CORPORATION

FIGURE 26b

ANALYST: ______________________________ PAGE: _____

DATE: ______________________________

F	I	E	L	D	Δ	D	E	S	I	G	N	
Δ	Δ	Δ	Δ	Δ	Δ	Δ	Δ	Δ	Δ	Δ	Δ	
P	R	O	D	U	C	T	-	N	O	Δ	Δ	Name of field.
0	0	0	0	0	0	0	0	0	0	0	0	Word location in item. [000 to (item size-1)]
0	0	0	0	0	2	ɨ	ɨ	ɨ	1	C	0	Field descriptor of form 00000TPPSLN0*
0	0	0	0	0	0	0	0	0	0	0	0	Extractor; if full-word field, all 0's.
Q	U	A	N	T	I	T	Y	Δ	Δ	Δ	Δ	
0	0	0	0	0	1	0	0	0	0	0	0	
0	0	0	0	0	3	ɨ	ɨ	ɨ	7	6	0	
0	0	0	0	0	0	1	1	1	1	1	1	
E	N	D	Δ	F	I	L	E	Δ	D	E	S	
0	0	0				0	0	0	0	0	0	
0	0	0	0	0							0	
		ZERO FILL THRU WORD 058										
0	0	0				0	0	0	0	0	0	
0	0	0	0	0							0	
0	0	0				0	0	0	0	0	0	
0	0	0	0	0							0	
0	0	0				0	0	0	0	0	0	
0	0	0	0	0							0	
0	0	0				0	0	0	0	0	0	
0	0	0	0	0							0	
E	N	D	Δ	F	I	L	E	Δ	D	E	S	

An unlimited number of fields may be described using the same four-word packet format.

* Explanation of field descriptor:

T	=	Type of field. 1 - alphabetic 2 - alpha-numeric 3 - numeric
PP	=	Position of decimal point in relation to a reference point immediately to the left of the left-most digit of the field. 00 - coincident with reference point ɨɨ - not applicable nL - n positions to the left of the reference point nR - n positions to the right of the reference point (n = 1,2,...,9,A,B,...,Z)
S	=	Digit position of the sign. 1,2,...,9,A,B, or C ɨ - not applicable
L	=	Digit position of the left-most digit of the field, excluding sign. 1,2,...,9,A,B, or C
N	=	Number of digits in the field, excluding sign. 1,2,...,9,A,B, or C (A = 10, B = 11, C = 12)

NOTE: Place the sentinel ENDΔFILEΔDES immediately following the last four-word packet and in word 059 of that block.

FLOW-MATIC DATA DESIGN FORM 3

FIGURE 26C

Remington Rand Univac
DIVISION OF SPERRY RAND CORPORATION

S1-1501

ANALYST: ______________________ PAGE: ______

DATE: ______________________

N	A	M	E	△	O	F	△	F	I	L	E	
P	R	I	C	E	△	△	△	△	△	△	△	English name of file.
F	I	L	E	△	D	E	S	I	G	N	△	
△	△	△	△	△	△	△	△	△	△	△	△	
L	A	B	E	L	△	△	△	△	△	△	△	
M	M	D	D	Y	Y	1	0	0	2	0	1	1.) Label with reel counter. 2.) If label variable all △'s.
L	O	C	△	O	F	△	L	A	B	E	L	
0	0	0	0	0	0	0	0	0	0	0	3	Word location of label in label block.
M	U	L	T	I	△	R	E	E	L	△	△	
0	0	0	0	0	0	0	0	0	0	0	0	1 = Yes; 0 = No.
B	L	K	△	C	T	△	I	N	D	△	△	
0	0	0	0	0	0	0	0	0	0	0	1	1 = Block count desired; 0 = No block count.
B	L	K	△	C	T	△	L	O	C	△	△	
0	0	0	0	0	0	0	0	0	0	0	1	Word location of block count in last item of sentinel block. [000 to (item size-1)]
E	N	D	△	R	E	E	L	△	S	E	N	
△	△	△	△	△	△	△	△	△	△	△	△	End of reel sentinel. If single reel, all △'s.
E	N	D	△	F	I	L	E	△	S	E	N	
Z	Z	Z	Z	Z	Z	Z	Z	Z	Z	Z	Z	End of file sentinel.
L	O	C	△	I	N	△	F	I	R	S	T	
0	0	0	0	0	0	0	0	0	0	0	0	Word location of sentinel in first invalid item. [000 to (item size-1)]
L	O	C	△	I	N	△	L	A	S	T	△	
0	0	0	0	0	0	0	0	0	0	0	0	Word location of sentinel in last item of sentinel block. [000 to (item size-1)]
												Other entries may be added here, each consisting of a title word and an information word; e.g., PART△BLK△SEN ZZZZZZZZZZZX
												UNITYPIST NOTE: After the last entry skip to the next page.
												FLOW-MATIC DATA DESIGN FORM 1
												FIGURE 26d Remington Rand Univac DIVISION OF SPERRY RAND CORPORATION

S1-1499

ANALYST: ______________________ PAGE: ____

DATE: ______________________

I	T	E	M	Δ	D	E	S	I	G	N	Δ	
Δ	Δ	Δ	Δ	Δ	Δ	Δ	Δ	Δ	Δ	Δ	Δ	
I	T	E	M	Δ	S	I	Z	E	Δ	Δ	Δ	
0	0	0	0	0	0	0	0	0	0	0	2	1,2,3,4,5,6,10,12,15,20,30,60.
N	O	Δ	O	F	Δ	K	E	Y	S	Δ	Δ	
0	0	0	0	0	0	0	0	0	0	0	1	0,1,2,....,9 = number of keys
K	E	Y	Δ	1	Δ	Δ	Δ	Δ	Δ	Δ	Δ	
P	R	O	D	U	C	T	-	N	O	Δ	Δ	Name of field, if no key ΔΔΔΔΔΔΔΔΔΔΔΔ

Further key entries may be added here, each consisting of KEYΔnΔΔΔΔΔΔ followed by the name of the field.

If there are Sub-items to be described, the descriptions are entered following the last Key entry. Sub-items are described with two-word packets consisting of the name of the Sub-item followed by a word in the format 000SSS000EEE, where SSS is the first word and EEE is the last word of the Sub-item, relative to the entire item.

UNITYPIST
<u>NOTE:</u> After the last entry skip to the next page.

FLOW-MATIC DATA DESIGN FORM 2

Remington Rand Univac
DIVISION OF SPERRY RAND CORPORATION

FIGURE 26e

S1-1500

ANALYST: ______________________ PAGE: ______

DATE: ______________________

F	I	E	L	D	Δ	D	E	S	I	G	N	
Δ	Δ	Δ	Δ	Δ	Δ	Δ	Δ	Δ	Δ	Δ	Δ	
P	R	O̲	D	U	C	T	-	N	O̲	Δ	Δ	Name of field.
0	0	0	0	0	0	0	0	0	0	0	0	Word location in item. [000 to (item size-1)]
0	0	0	0	0	2	ɨ	ɨ	ɨ	1	C	0	Field descriptor of form 00000TPPSLN0*
0	0	0	0	0	0	0	0	0	0	0	0	Extractor; if full-word field, all 0's.
U	N	I	T	-	P	R	I	C	E	Δ	Δ	
0	0	0	0	0	1	0	0	0	0	0	0	
0	0	0	0	0	3	3	R	ɨ	8	5	0	
0	0	0	0	0	0	0	1	1	1	1	1	
E	N	D	Δ	F	I	L	E	Δ	D	E	S	
0	0	0				0	0	0	0	0	0	
0	0	0	0								0	
		ZERO FILL THRU WORD 058										
0	0	0				0	0	0	0	0	0	
0	0	0	0	0							0	
0	0	0				0	0	0	0	0	0	
0	0	0	0	0							0	
0	0	0				0	0	0	0	0	0	
0	0	0	0	0							0	
0	0	0				0	0	0	0	0	0	
0	0	0	0	0							0	
E	N	D	Δ	F	I	L	E	Δ	D	E	S	

An unlimited number of fields may be described using the same four-word packet format.

* Explanation of field descriptor:

T = Type of field.
- 1 - alphabetic
- 2 - alpha-numeric
- 3 - numeric

PP = Position of decimal point in relation to a reference point immediately to the left of the left-most digit of the field.
- 00 - coincident with reference point
- ɨɨ - not applicable
- nL - n positions to the left of the reference point
- nR - n positions to the right of the reference point
- (n = 1,2,...,9,A,B,...,Z)

S = Digit position of the sign.
- 1,2,...,9,A,B, or C
- ɨ - not applicable

L = Digit position of the left-most digit of the field, excluding sign.
- 1,2,...,9,A,B, or C

N = Number of digits in the field, excluding sign.
- 1,2,...,9,A,B, or C
- (A = 10, B = 11, C = 12)

NOTE: Place the sentinel ENDΔFILEΔDES immediately following the last four-word packet and in word 059 of that block.

FLOW-MATIC DATA DESIGN FORM 3

Remington Rand Univac
DIVISION OF SPERRY RAND CORPORATION

FIGURE 26f

S1-1501

ANALYST: ______________________ PAGE: ______

DATE: ______________________

N	A	M	E	Δ	O	F	Δ	F	I	L	E	
P	R	I	C	E	D	-	I	N	V	Δ	Δ	English name of file.
F	I	L	E	Δ	D	E	S	I	G	N	Δ	
Δ	Δ	Δ	Δ	Δ	Δ	Δ	Δ	Δ	Δ	Δ	Δ	
L	A	B	E	L	Δ	Δ	Δ	Δ	Δ	Δ	Δ	
M	M	D	D	Y	Y	I	0	0	3	0	I	1.) Label with reel counter. 2.) If label variable all Δ's.
L	O	C	Δ	O	F	Δ	L	A	B	E	L	
0	0	0	0	0	0	0	0	0	0	0	3	Word location of label in label block.
M	U	L	T	I	Δ	R	E	E	L	Δ	Δ	
0	0	0	0	0	0	0	0	0	0	0	I	1 = Yes; 0 = No.
B	L	K	Δ	C	T	Δ	I	N	D	Δ	Δ	
0	0	0	0	0	0	0	0	0	0	0	I	1 = Block count desired; 0 = No block count.
B	L	K	Δ	C	T	Δ	L	O	C	Δ	Δ	
0	0	0	0	0	0	0	0	0	0	0	I	Word location of block count in last item of sentinel block. [000 to (item size-1)]
E	N	D	Δ	R	E	E	L	Δ	S	E	N	
Z	Z	Z	Z	Z	Z	Z	Z	Z	Z	Z	Y	End of reel sentinel. If single reel, all Δ's.
E	N	D	Δ	F	I	L	E	Δ	S	E	N	
Z	Z	Z	Z	Z	Z	Z	Z	Z	Z	Z	Z	End of file sentinel.
L	O	C	Δ	I	N	Δ	F	I	R	S	T	
0	0	0	0	0	0	0	0	0	0	0	0	Word location of sentinel in first invalid item. [000 to (item size-1)]
L	O	C	Δ	I	N	Δ	L	A	S	T	Δ	
0	0	0	0	0	0	0	0	0	0	0	0	Word location of sentinel in last item of sentinel block. [000 to (item size-1)]

Other entries may be added here, each consisting of a title word and an information word; e.g., PARTΔBLKΔSEN
ZZZZZZZZZZZX

UNITYPIST
NOTE: After the last entry skip to the next page.

FLOW-MATIC DATA DESIGN FORM 1

Remington Rand Univac
DIVISION OF SPERRY RAND CORPORATION

FIGURE 26g

ANALYST: ______________________ PAGE: ____

DATE: ______________________

I	T	E	M	Δ	D	E	S	I	G	N	Δ	
Δ	Δ	Δ	Δ	Δ	Δ	Δ	Δ	Δ	Δ	Δ	Δ	
I	T	E	M	Δ	S	I	Z	E	Δ	Δ	Δ	
0	0	0	0	0	0	0	0	0	0	1	0	1,2,3,4,5,6,10,12,15,20,30,60.
N	O̲	Δ	O̲	F	Δ	K	E	Y	S	Δ	Δ	
0	0	0	0	0	0	0	0	0	0	0	1	0,1,2,....,9 = number of keys
K	E	Y	Δ	1	Δ	Δ	Δ	Δ	Δ	Δ	Δ	
P	R	O̲	D	U	C	T	-	N	O̲	Δ	Δ	Name of field, if no key ΔΔΔΔΔΔΔΔΔΔΔΔ

Further key entries may be added here, each consisting of KEYΔnΔΔΔΔΔΔ followed by the name of the field.

If there are Sub-items to be described, the descriptions are entered following the last Key entry. Sub-items are described with two-word packets consisting of the name of the Sub-item followed by a word in the format 000SSS000EEE, where SSS is the first word and EEE is the last word of the Sub-item, relative to the entire item.

UNITYPIST
NOTE: After the last entry skip to the next page.

FLOW-MATIC DATA DESIGN FORM 2

Remington Rand Univac
DIVISION OF SPERRY RAND CORPORATION

FIGURE 26h

S1-1500

ANALYST: PAGE:

DATE:

F	I	E	L	D	Δ	D	E	S	I	G	N	
Δ	Δ	Δ	Δ	Δ	Δ	Δ	Δ	Δ	Δ	Δ	Δ	
P	R	O	D	U	C	T	-	N	O	Δ	Δ	Name of field.
0	0	0	0	0	0	0	0	0	0	0	0	Word location in item. [000 to (item size-1)]
0	0	0	0	0	2	ɨ	ɨ	ɨ	1	C	0	Field descriptor of form 00000TPPSLN0*
0	0	0	0	0	0	0	0	0	0	0	0	Extractor; if full-word field, all 0's.
Q	U	A	N	T	I	T	Y	Δ	Δ	Δ	Δ	
0	0	0	0	0	1	0	0	0	0	0	0	
0	0	0	0	0	3	ɨ	ɨ	ɨ	7	6	0	
0	0	0	0	0	0	1	1	1	1	1	1	
U	N	I	T	-	P	R	I	C	E	Δ	Δ	
0	0	0	0	0	2	0	0	0	0	0	0	
0	0	0	0	0	3	3	R	ɨ	8	5	0	
0	0	0	0	0	0	0	1	1	1	1	1	
E	X	T	-	P	R	I	C	E	Δ	Δ	Δ	
0	0	0	0	0	3	0	0	0	0	0	0	
0	0	0	0	0	3	8	R	ɨ	3	A	0	
0	0	1	1	1	1	1	1	1	1	1	1	
E	N	D	Δ	F	I	L	E	Δ	D	E	S	
0	0	0				0	0	0	0	0	0	
0	0	0	0								0	
		ZERO FILL THRU WORD 058										
0	0	0				0	0	0	0	0	0	
0	0	0	0	0							0	
0	0	0				0	0	0	0	0	0	
0	0	0	0	0							0	
E	N	D	Δ	F	I	L	E	Δ	D	E	S	

An unlimited number of fields may be described using the same four-word packet format.

* Explanation of field descriptor:

T = Type of field.
- 1 - alphabetic
- 2 - alpha-numeric
- 3 - numeric

PP = Position of decimal point in relation to a reference point immediately to the left of the left-most digit of the field.
- 00 - coincident with reference point
- ɨɨ - not applicable
- nL - n positions to the left of the reference point
- nR - n positions to the right of the reference point (n = 1,2,...,9,A,B,...,Z)

S = Digit position of the sign.
- 1,2,...,9,A,B, or C
- ɨ - not applicable

L = Digit position of the left-most digit of the field, excluding sign.
- 1,2,...,9,A,B, or C

N = Number of digits in the field, excluding sign.
- 1,2,...,9,A,B, or C
- (A = 10, B = 11, C = 12)

<u>NOTE</u>: Place the sentinel ENDΔFILEΔDES immediately following the last four-word packet <u>and</u> in word 059 of that block.

FLOW-MATIC DATA DESIGN FORM 3

Remington Rand Univac
DIVISION OF SPERRY RAND CORPORATION

FIGURE 26i

S1-1501

ANALYST: ____________________ PAGE: ______

DATE: ____________________

N	A	M	E	Δ	O	F	Δ	F	I	L	E	
U	N	P	R	I	C	E	D	-	I	N	V	English name of file.
F	I	L	E	Δ	D	E	S	I	G	N	Δ	
Δ	Δ	Δ	Δ	Δ	Δ	Δ	Δ	Δ	Δ	Δ	Δ	
L	A	B	E	L	Δ	Δ	Δ	Δ	Δ	Δ	Δ	
M	M	D	D	Y	Y	I	0	0	4	0	I	1.) Label with reel counter. 2.) If label variable all Δ's.
L	O	C	Δ	O	F	Δ	L	A	B	E	L	
0	0	0	0	0	0	0	0	0	0	0	3	Word location of label in label block.
M	U	L	T	I	Δ	R	E	E	L	Δ	Δ	
0	0	0	0	0	0	0	0	0	0	0	I	1 = Yes; 0 = No.
B	L	K	Δ	C	T	Δ	I	N	D	Δ	Δ	
0	0	0	0	0	0	0	0	0	0	0	I	1 = Block count desired; 0 = No block count.
B	L	K	Δ	C	T	Δ	L	O	C	Δ	Δ	
0	0	0	0	0	0	0	0	0	0	0	I	Word location of block count in last item of sentinel block. [000 to (item size-1)]
E	N	D	Δ	R	E	E	L	Δ	S	E	N	
Z	Z	Z	Z	Z	Z	Z	Z	Z	Z	Z	Y	End of reel sentinel. If single reel, all Δ's.
E	N	D	Δ	F	I	L	E	Δ	S	E	N	
Z	Z	Z	Z	Z	Z	Z	Z	Z	Z	Z	Z	End of file sentinel.
L	O	C	Δ	I	N	Δ	F	I	R	S	T	
0	0	0	0	0	0	0	0	0	0	0	0	Word location of sentinel in first invalid item. [000 to (item size-1)]
L	O	C	Δ	I	N	Δ	L	A	S	T	Δ	
0	0	0	0	0	0	0	0	0	0	0	0	Word location of sentinel in last item of sentinel block. [000 to (item size-1)]

Other entries may be added here, each consisting of a title word and an information word; e.g., PARTΔBLKΔSEN
ZZZZZZZZZZZX

UNITYPIST
NOTE: After the last entry skip to the next page.

FLOW-MATIC DATA DESIGN FORM 1

Remington Rand Univac
DIVISION OF SPERRY RAND CORPORATION

FIGURE 26j

S1-1499

ANALYST: ______________________ PAGE: ____

DATE: ______________________

I	T	E	M	Δ	D	E	S	I	G	N	Δ	
Δ	Δ	Δ	Δ	Δ	Δ	Δ	Δ	Δ	Δ	Δ	Δ	
I	T	E	M	Δ	S	I	Z	E	Δ	Δ	Δ	
0	0	0	0	0	0	0	0	0	0	1	0	1,2,3,4,5,6,10,12,15,20,30,60.
N	O	Δ	O	F	Δ	K	E	Y	S	Δ	Δ	
0	0	0	0	0	0	0	0	0	0	0	1	0,1,2,....,9 = number of keys
K	E	Y	Δ	1	Δ	Δ	Δ	Δ	Δ	Δ	Δ	
P	R	O	D	U	C	T	-	N	O	Δ	Δ	Name of field, if no key ΔΔΔΔΔΔΔΔΔΔΔΔ

Further key entries may be added here, each consisting of KEYΔnΔΔΔΔΔΔΔ followed by the name of the field.

If there are Sub-items to be described, the descriptions are entered following the last Key entry. Sub-items are described with two-word packets consisting of the name of the Sub-item followed by a word in the format 000SSS000EEE, where SSS is the first word and EEE is the last word of the Sub-item, relative to the entire item.

UNITYPIST
<u>NOTE:</u> After the last entry skip to the next page.

FLOW-MATIC DATA DESIGN FORM 2

Remington Rand Univac
DIVISION OF SPERRY RAND CORPORATION

FIGURE 26k

S1-1500

ANALYST: ______________________ PAGE: ____

DATE: ______________________

F	I	E	L	D	Δ	D	E	S	I	G	N	
Δ	Δ	Δ	Δ	Δ	Δ	Δ	Δ	Δ	Δ	Δ	Δ	
P	R	O	D	U	C	T	-	N	O	Δ	Δ	Name of field.
0	0	0	0	0	0	0	0	0	0	0	0	Word location in item. [000 to (item size-1)]
0	0	0	0	0	2	ɨ	ɨ	ɨ	1	C	0	Field descriptor of form 00000TPPSLN0*
0	0	0	0	0	0	0	0	0	0	0	0	Extractor; if full-word field, all 0's.
Q	U	A	N	T	I	T	Y	Δ	Δ	Δ	Δ	
0	0	0	0	0	1	0	0	0	0	0	0	
0	0	0	0	0	3	ɨ	ɨ	ɨ	7	6	0	
0	0	0	0	0	0	1	1	1	1	1	1	
E	N	D	Δ	F	I	L	E	Δ	D	E	S	
0	0	0				0	0	0	0	0	0	
0	0	0	0	0							0	
		ZERO FILL THRU WORD 058										
0	0	0				0	0	0	0	0	0	
0	0	0	0	0							0	
0	0	0				0	0	0	0	0	0	
0	0	0	0	0							0	
0	0	0				0	0	0	0	0	0	
0	0	0	0	0							0	
0	0	0				0	0	0	0	0	0	
0	0	0	0	0							0	
E	N	D	Δ	F	I	L	E	Δ	D	E	S	

An unlimited number of fields may be described using the same four-word packet format.

* Explanation of field descriptor:

T = Type of field.
- 1 - alphabetic
- 2 - alpha-numeric
- 3 - numeric

PP = Position of decimal point in relation to a reference point immediately to the left of the left-most digit of the field.
- 00 - coincident with reference point
- ɨɨ - not applicable
- nL - n positions to the left of the reference point
- nR - n positions to the right of the reference point

(n = 1,2,...,9,A,B,...,Z)

S = Digit position of the sign.
- 1,2,...,9,A,B, or C
- ɨ - not applicable

L = Digit position of the left-most digit of the field, excluding sign.
- 1,2,...,9,A,B, or C

N = Number of digits in the field, excluding sign.
- 1,2,...,9,A,B, or C
- (A = 10, B = 11, C = 12)

NOTE: Place the sentinel ENDΔFILEΔDES immediately following the last four-word packet and in word 059 of that block.

FLOW-MATIC DATA DESIGN FORM 3

FIGURE 261

Remington Rand Univac
DIVISION OF SPERRY RAND CORPORATION

DATA DESIGN INPUT FOR SAMPLE PROBLEM I

```
NAME OF FILE  INVENTORY     FILE DESIGN                 LABEL         MMDDYY100101  LOC OF LABEL  000000000003  MULTI REEL    000000000001
BLK CT IND    000000000001  BLK CT LOC    000000000001  END REEL SEN  ZZZZZZZZZZZY  END FILE SEN  ZZZZZZZZZZZZ  LOC IN FIRST  000000000000
LOC IN LAST   000000000000  ITEM DESIGN                 ITEM SIZE     000000000010  NO OF KEYS    000000000001  KEY 1         PRODUCT-NO
FIELD DESIGN                PRODUCT-NO    000000000000  000002   ICO  000000000000  QUANTITY      000001000000  000003   760  000000111111
END FILE DES  000000000000  000000000000  000000000000  000000000000  000000000000  000000000000  000000000000  000000000000  000000000000
000000000000  000000000000  000000000000  000000000000  000000000000  000000000000  000000000000  000000000000  000000000000  END FILE DES


NAME OF FILE  PRICE         FILE DESIGN                 LABEL         MMDDYY100201  LOC OF LABEL  000000000003  MULTI REEL    000000000000
BLK CT IND    000000000001  BLK CT LOC    000000000001  END REEL SEN                END FILE SEN  ZZZZZZZZZZZZ  LOC IN FIRST  000000000000
LOC IN LAST   000000000000  ITEM DESIGN                 ITEM SIZE     000000000002  NO OF KEYS    000000000001  KEY 1         PRODUCT-NO
FIELD DESIGN                PRODUCT-NO    000000000000  000002   ICO  000000000000  UNIT-PRICE    000001000000  0000033R 850  000000011111
END FILE DES  000000000000  000000000000  000000000000  000000000000  000000000000  000000000000  000000000000  000000000000  000000000000
000000000000  000000000000  000000000000  000000000000  000000000000  000000000000  000000000000  000000000000  000000000000  END FILE DES


NAME OF FILE  PRICED-INV    FILE DESIGN                 LABEL         MMDDYY100301  LOC OF LABEL  000000000003  MULTI REEL    000000000001
BLK CT IND    000000000001  BLK CT LOC    000000000001  END REEL SEN  ZZZZZZZZZZZY  END FILE SEN  ZZZZZZZZZZZZ  LOC IN FIRST  000000000000
LOC IN LAST   000000000000  ITEM DESIGN                 ITEM SIZE     000000000010  NO OF KEYS    000000000001  KEY 1         PRODUCT-NO
FIELD DESIGN                PRODUCT-NO    000000000000  000002   ICO  000000000000  QUANTITY      000001000000  000003   760  000000111111
UNIT-PRICE    000002000000  0000033R 850  000000011111  EXT-PRICE     000003000000  0000038R 3A0  001111111111  END FILE DES  000000000000
000000000000  000000000000  000000000000  000000000000  000000000000  000000000000  000000000000  000000000000  000000000000  END FILE DES


NAME OF FILE  UNPRICED-INV  FILE DESIGN                 LABEL         MMDDYY100401  LOC OF LABEL  000000000003  MULTI REEL    000000000001
BLK CT IND    000000000001  BLK CT LOC    000000000001  END REEL SEN  ZZZZZZZZZZZY  END FILE SEN  ZZZZZZZZZZZZ  LOC IN FIRST  000000000000
LOC IN LAST   000000000000  ITEM DESIGN                 ITEM SIZE     000000000010  NO OF KEYS    000000000001  KEY 1         PRODUCT-NO
FIELD DESIGN                PRODUCT-NO    000000000000  000002   ICO  000000000000  QUANTITY      000001000000  000003   760  000000111111
END FILE DES  000000000000  000000000000  000000000000  000000000000  000000000000  000000000000  000000000000  000000000000  000000000000
000000000000  000000000000  000000000000  000000000000  000000000000  000000000000  000000000000  000000000000  000000000000  END FILE DES
```

FIGURE 27a

FLOWMATIC CODE INPUT FOR SAMPLE PROBLEM I

```
 (0) INPUT INVENTORY FILE-A PRICE FILE-B ; OUTPUT PRICED-INV FILE-C UNPRICED-INV FILE-D ; HSP D .
 (1) COMPARE PRODUCT-NO (A) WITH PRODUCT-NO (B) ; IF GREATER GO TO OPERATION 10 ; IF EQUAL GO TO OPERATION 5 ; OTHERWISE
GO TO OPERATION 2 .
 (2) TRANSFER A TO D .
 (3) WRITE-ITEM D .
 (4) JUMP TO OPERATION 8 .
 (5) TRANSFER A TO C .
 (6) MOVE UNIT-PRICE (B) TO UNIT-PRICE (C)
 (7) WRITE-ITEM C .
 (8) READ-ITEM A ; IF END OF DATA GO TO OPERATION 14 .
 (9) JUMP TO OPERATION 1 .
(10) READ-ITEM B ; IF END OF DATA GO TO OPERATION 12 .
(11) JUMP TO OPERATION 1 .
(12) SET OPERATION 9 TO GO TO OPERATION 2 .
(13) JUMP TO OPERATION 2 .
(14) TEST PRODUCT-NO (B) AGAINST ZZZZZZZZZZZZ ; IF EQUAL GO TO OPERATION 16 ; OTHERWISE GO TO OPERATION 15 .
(15) REWIND B .
(16) CLOSE-OUT FILE C , D .
(17) STOP . (END)

ZZZZZZZZZZZZ000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000
000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000
000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000
000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000
000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000
000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000ZZZZZZZZZZZZ
```

FIGURE 27b

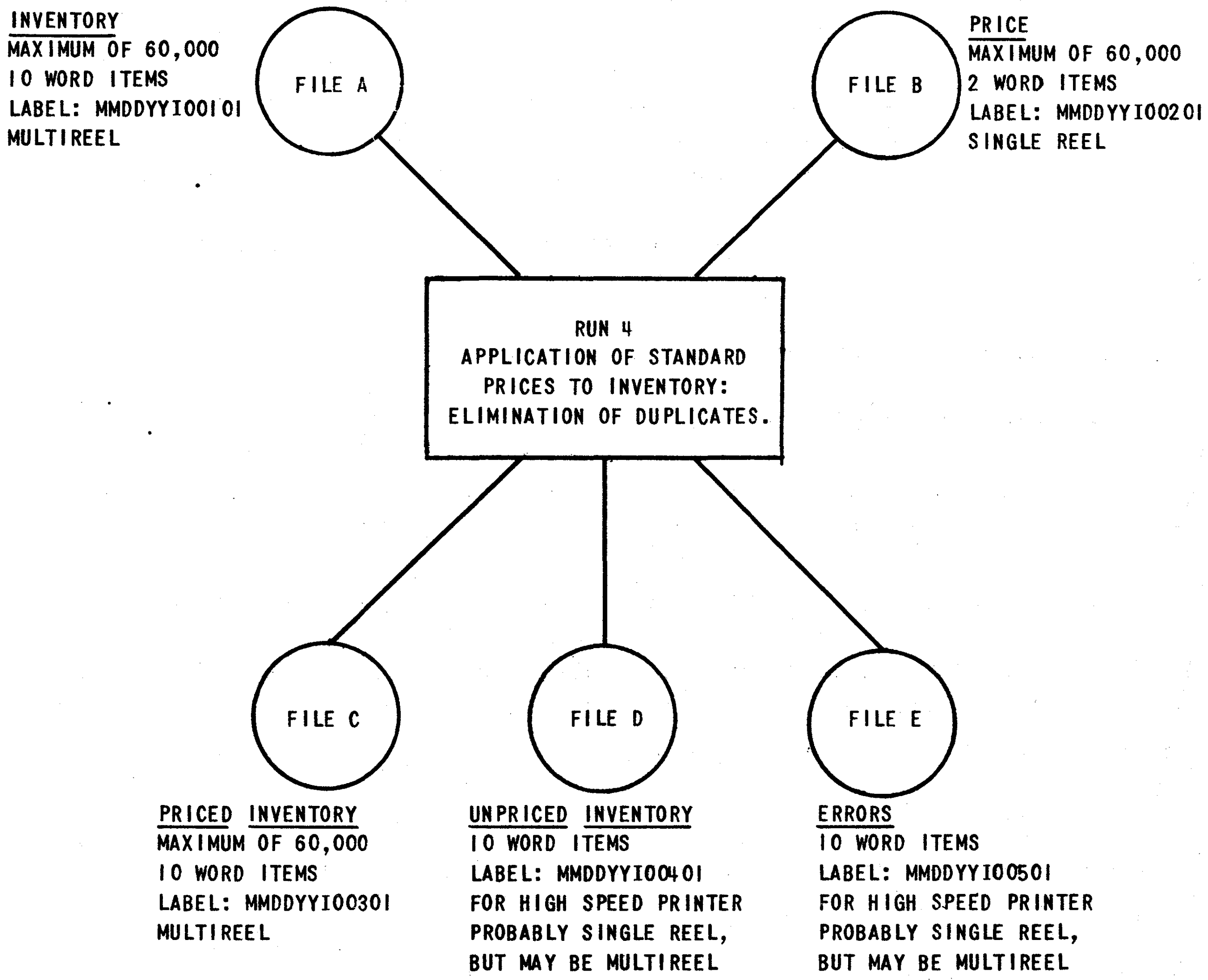
chapter 5

Intermediate or Working Storage

In order to introduce a new concept, consider a simple variation in the sample problem. Suppose that the definition is altered by the fact that duplicate product numbers may occur in the Inventory file, and that such additional items are the result of errors in the original creation of the file. Suppose further that these extra items are to be placed in an Error output file for printing.

The process chart of this *second* sample problem, with details of the formats of the files, is shown in Figure 28.

PROCESS CHART
ABC MANUFACTURING COMPANY INVENTORY PROBLEM 2



Conventions

(1) LABELS IN WORD 03 OF FIRST BLOCK ON EACH REEL.

(2) BLOCK COUNTS IN WORD 01 OF LAST ITEM IN SENTINEL BLOCK.

(3) SENTINELS ARE ZZZZZZZZZZZY FOR END OF REEL AND ZZZZZZZZZZZZ FOR END OF FILE. THESE ARE LOCATED IN THE KEY WORD POSITION (WORD 000) OF FIRST INVALID ITEM AND LAST ITEM OF SENTINEL BLOCK.

FIGURE 28

An outline of the logical steps to be performed can be written as follows:

Assume at the start that the first item of each of the input files is available.

I Compare the product number of the Inventory item with the product number of the Price item.

  a. If the Inventory product number is less, go on to step II.
  b. If the product numbers are equal, go on to step III.
  c. If the Price product number is less, go on to step VII.

II Prepare and write an Unpriced Inventory item. Then go on to step IV.

III Prepare and write a Priced Inventory item.

IV Store the product number.

V Read the next Inventory item and go on to step VI. Or, if the Inventory file is exhausted, wind up the problem and stop.

VI Compare the new product number with the stored product number.

  a. If the product numbers are equal, a duplicate has been found. Prepare and write an Error item. Then go to step V.
  b. If the product numbers are not equal, go back to step I.

VII Read the next Price item, and go back to step I.

Note in particular step VII. If the Price file reaches end of data before the Inventory file, there is no need to eliminate the unnecessary comparison. This is true because the sentinel at the end of the Price file is in the same field as the product number, and the sentinel is larger than any legitimate product number. Figure 16 on page 19 illustrated the method used to eliminate the unnecessary comparison.

This statement of the problem can be better shown on block chart form (Figure 29).

BLOCK CHART

ABC MANUFACTURING COMPANY INVENTORY - PROBLEM 2

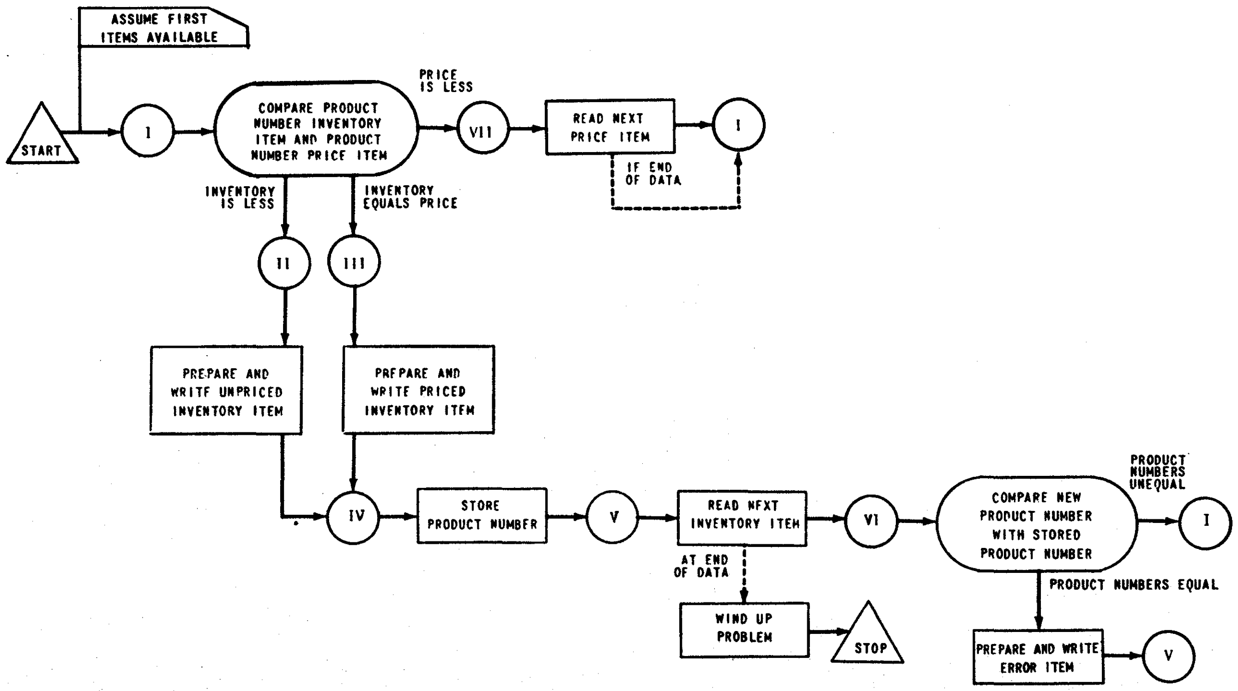

ASSUME FIRST ITEMS AVAILABLE
START
I
COMPARE PRODUCT NUMBER INVENTORY ITEM AND PRODUCT NUMBER PRICE ITEM
PRICE IS LESS
VII
READ NEXT PRICE ITEM
I
IF END OF DATA
INVENTORY IS LESS
INVENTORY EQUALS PRICE
II
III
PREPARE AND WRITE UNPRICED INVENTORY ITEM
PREPARE AND WRITE PRICED INVENTORY ITEM
IV
STORE PRODUCT NUMBER
V
READ NEXT INVENTORY ITEM
VI
COMPARE NEW PRODUCT NUMBER WITH STORED PRODUCT NUMBER
PRODUCT NUMBERS UNEQUAL
I
AT END OF DATA
WIND UP PROBLEM
STOP
PRODUCT NUMBERS EQUAL
PREPARE AND WRITE ERROR ITEM
V

FIGURE 29

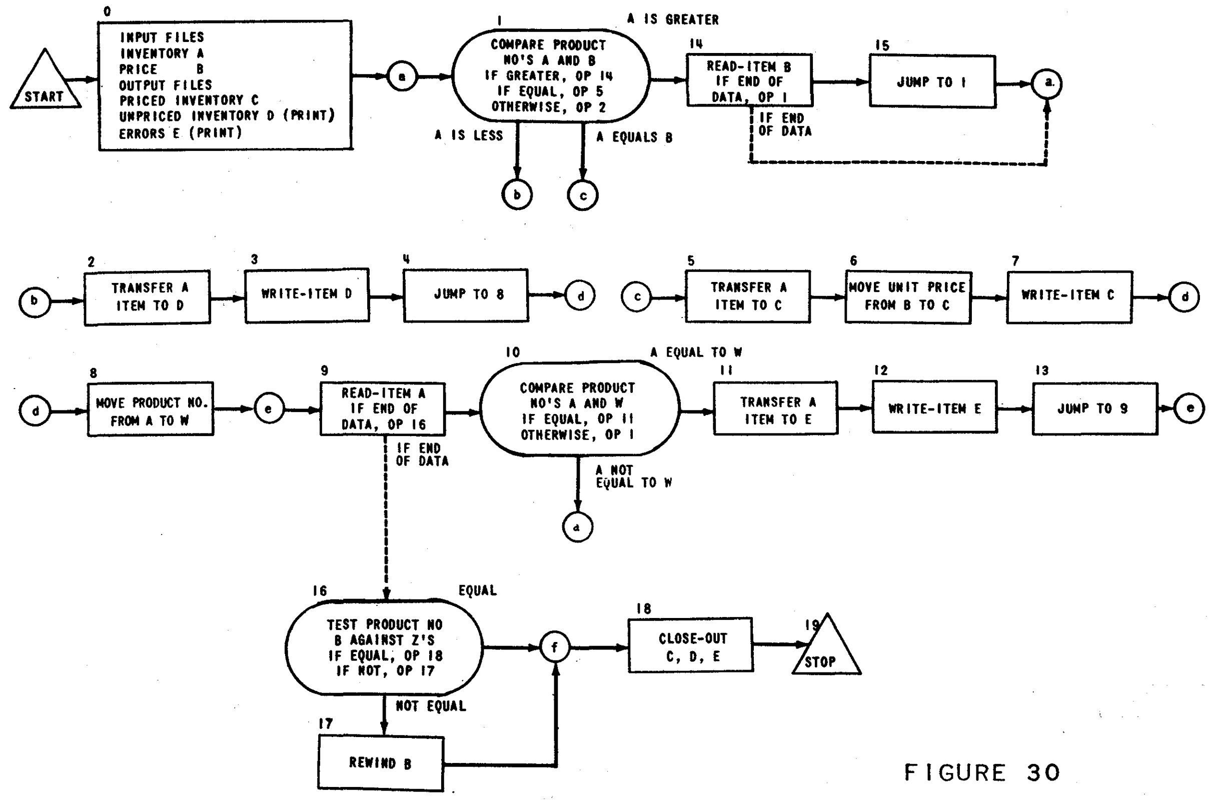
In Sample Problem 1, no information had to be set aside for further reference; all information could be acted upon at the time it became available. The new concept introduced here in Sample Problem 2 is that of *saving* information for future use. A common term for that part of the computer storage reserved for this purpose in a given problem is Working Storage. In *FLOW-MATIC* the term is abbreviated to W-storage. The programmer may save as many fields or items as desired in W-storage. And he may name these fields in the same way that English names are assigned to fields within the data files. It follows, that, when referring to fields in W-storage, he labels them with the letter W in parentheses, e.g., Operations (8) and (10) on page 65.

The *FLOW-MATIC* chart for Sample Problem 2 in Figure 30 is created in the same general manner as before, but makes use of this additional facility. The development of the *FLOW-MATIC* code also follows the previous methods. Note however, the use of the letter W in Figure 31, operations eight and ten.

It remains to write the Data Designs for sample problem 2. Clearly the Data Designs for files A, B, C, and D need not be changed. The Data Design for the new file, E, is shown in Figures 32 on pages 66 and 67. Note that, since no field names within the file are required in the *FLOW-MATIC* code, none have been described in the Data Design.

FLOW-MATIC CHART

ABC MANUFACTURING COMPANY INVENTORY - PROBLEM 2



START
0 INPUT FILES INVENTORY A PRICE B OUTPUT FILES PRICED INVENTORY C UNPRICED INVENTORY D (PRINT) ERRORS E (PRINT)
1 COMPARE PRODUCT NO'S A AND B IF GREATER, OP 14 IF EQUAL, OP 5 OTHERWISE, OP 2
A IS GREATER
A IS LESS
A EQUALS B
14 READ-ITEM B IF END OF DATA, OP 1
IF END OF DATA
15 JUMP TO 1
2 TRANSFER A ITEM TO D
3 WRITE-ITEM D
4 JUMP TO 8
5 TRANSFER A ITEM TO C
6 MOVE UNIT PRICE FROM B TO C
7 WRITE-ITEM C
8 MOVE PRODUCT NO. FROM A TO W
9 READ-ITEM A IF END OF DATA, OP 16
IF END OF DATA
10 COMPARE PRODUCT NO'S A AND W IF EQUAL, OP 11 OTHERWISE, OP 1
A EQUAL TO W
A NOT EQUAL TO W
11 TRANSFER A ITEM TO E
12 WRITE-ITEM E
13 JUMP TO 9
16 TEST PRODUCT NO B AGAINST Z'S IF EQUAL, OP 18 IF NOT, OP 17
EQUAL
NOT EQUAL
17 REWIND B
18 CLOSE-OUT C, D, E
19 STOP

FIGURE 30

FLOW-MATIC CODE

```
(0)  INPUT INVENTORY FILE-A PRICE FILE-B ; OUTPUT PRICED-INV FILE-C UNPRICED-
     INV FILE-D ERROR FILE-E ; HSP D , E .
(1)  COMPARE PRODUCT-NO (A) WITH PRODUCT-NO (B) ; IF GREATER GO TO OPERATION
     14 ; IF EQUAL GO TO OPERATION 5 ; OTHERWISE GO TO OPERATION 2 .
(2)  TRANSFER A TO D .
(3)  WRITE-ITEM D .
(4)  JUMP TO OPERATION 8 .
(5)  TRANSFER A TO C .
(6)  MOVE UNIT-PRICE (B) TO UNIT-PRICE (C) .
(7)  WRITE-ITEM C .
(8)  MOVE PRODUCT-NO (A) TO PRODUCT-NO (W) .
(9)  READ-ITEM A ; IF END OF DATA GO TO OPERATION 16 .
(10) COMPARE PRODUCT-NO (A) WITH PRODUCT-NO (W) ; IF EQUAL GO TO OPERATION 11 ;
     OTHERWISE GO TO OPERATION 1 .
(11) TRANSFER A TO E .
(12) WRITE-ITEM E .
(13) JUMP TO OPERATION 9 .
(14) READ-ITEM B ; IF END OF DATA GO TO OPERATION 1 .
(15) JUMP TO OPERATION 1 .
(16) TEST PRODUCT-NO (B) AGAINST ZZZZZZZZZZZZ ; IF EQUAL GO TO OPERATION 18 ;
     OTHERWISE GO TO OPERATION 17 .
(17) REWIND B .
(18) CLOSE-OUT FILES C , D , E .
(19) STOP . (END)
```

ABC MANUFACTURING COMPANY INVENTORY PROBLEM 2

FIGURE 31

ANALYST: ______________ DATA DESIGN FOR ERROR OUTPUT FILE PAGE: ______

DATE: ______________ ABC MANUFACTURING COMPANY INVENTORY PROBLEM 2

N	A	M	E	Δ	O	F	Δ	F	I	L	E	
E	R	R	O	R	Δ	Δ	Δ	Δ	Δ	Δ	Δ	English name of file.
F	I	L	E	Δ	D	E	S	I	G	N	Δ	
Δ	Δ	Δ	Δ	Δ	Δ	Δ	Δ	Δ	Δ	Δ	Δ	
L	A	B	E	L	Δ	Δ	Δ	Δ	Δ	Δ	Δ	
M	M	D	D	Y	Y	1	0	0	5	0	1	1.) Label with reel counter. 2.) If label variable all Δ's.
L	O	C	Δ	O	F	Δ	L	A	B	E	L	
0	0	0	0	0	0	0	0	0	0	0	3	Word location of label in label block.
M	U	L	T	I	Δ	R	E	E	L	Δ	Δ	
0	0	0	0	0	0	0	0	0	0	0	1	1 = Yes; 0 = No.
B	L	K	Δ	C	T	Δ	I	N	D	Δ	Δ	
0	0	0	0	0	0	0	0	0	0	0	1	1 = Block count desired; 0 = No block count.
B	L	K	Δ	C	T	Δ	L	O	C	Δ	Δ	
0	0	0	0	0	0	0	0	0	0	0	1	Word location of block count in last item of sentinel block. [000 to (item size-1)]
E	N	D	Δ	R	E	E	L	Δ	S	E	N	
Z	Z	Z	Z	Z	Z	Z	Z	Z	Z	Z	Y	End of reel sentinel. If single reel, all Δ's.
E	N	D	Δ	F	I	L	E	Δ	S	E	N	
Z	Z	Z	Z	Z	Z	Z	Z	Z	Z	Z	Z	End of file sentinel.
L	O	C	Δ	I	N	Δ	F	I	R	S	T	
0	0	0	0	0	0	0	0	0	0	0	0	Word location of sentinel in first invalid item. [000 to (item size-1)]
L	O	C	Δ	I	N	Δ	L	A	S	T	Δ	
0	0	0	0	0	0	0	0	0	0	0	0	Word location of sentinel in last item of sentinel block. [000 to (item size-1)]

Other entries may be added here, each consisting of a title word and an information word; e.g., PARTΔBLKΔSEN
ZZZZZZZZZZZX

UNITYPIST
NOTE: After the last entry skip to the next page.

FLOW-MATIC DATA DESIGN FORM 1

FIGURE 32a

Remington Rand Univac
DIVISION OF SPERRY RAND CORPORATION

S1-1499

ANALYST: ____________________

DATE: ____________________

I	T	E	M	Δ	D	E	S	I	G	N	Δ	
Δ	Δ	Δ	Δ	Δ	Δ	Δ	Δ	Δ	Δ	Δ	Δ	
I	T	E	M	Δ	S	I	Z	E	Δ	Δ	Δ	
0	0	0	0	0	0	0	0	0	0	1	0	1,2,3,4,5,6,10,12,15,20,30,60.
N	O	Δ	O	F	Δ	K	E	Y	S	Δ	Δ	
0	0	0	0	0	0	0	0	0	0	0	0	0,1,2,....,9 = number of keys
K	E	Y	Δ	1	Δ	Δ	Δ	Δ	Δ	Δ	Δ	
Δ	Δ	Δ	Δ	Δ	Δ	Δ	Δ	Δ	Δ	Δ	Δ	Name of field, if no key ΔΔΔΔΔΔΔΔΔΔΔΔ
E	N	D	Δ	F	I	L	E	Δ	D	E	S	
			ZERO	FILL	THRU	WORD	058					
E	N	D	Δ	F	I	L	E	Δ	D	E	S	FIGURE 32b

Further key entries may be added here, each consisting of KEYΔnΔΔΔΔΔΔ followed by the name of the field.

If there are Sub-items to be described, the descriptions are entered following the last Key entry. Sub-items are described with two-word packets consisting of the name of the Sub-item followed by a word in the format 000SSS000EEE, where SSS is the first word and EEE is the last word of the Sub-item, relative to the entire item.

UNITYPIST

NOTE: After the last entry skip to the next page.

FLOW-MATIC DATA DESIGN FORM2

Remington Rand Univac

DIVISION OF SPERRY RAND CORPORATION

S1-1500

Now, however, there is a new kind of data to be described to *FLOW-MATIC,* namely, that assigned by the programmer to W-storage. This data is described to the compiler as if it were data from a file, except that none of the information pertaining to the organization of a tape is applicable. In Sample Problem 2, in fact, the only necessary description is of a single field, the product number. Therefore, no item design information is written, and under the field design section only one name is listed. The information required for this problem is shown below.

```
00        NAMEΔOFΔFILE    }   Heading
01        W-STORAGEΔΔΔ    }
02        FIELDΔDESIGN    }   Sub-Heading
03        ΔΔΔΔΔΔΔΔΔΔΔΔ    }
04        PRODUCT-NOΔΔ    |
05        000000000000    |   4 word Field
06        000002///ICO    |   Description
07        000000000000    |
08        ENDΔFILEΔDES
 .                        |
 :          Zero fill     |   Sentinels
 .                        |
59        ENDΔFILEΔDES
```

This Data Design is placed on the input tape for *FLOW-MATIC* together with other Data Designs. It will not be stored for use in other runs, since it is designed specifically for one program.

In addition, it is necessary to inform the compiler that some W-storage has been reserved by the programmer. This is done by writing a Directory for inclusion on the *FLOW-MATIC* input tape.

The Directory is a list showing the number of words in the programmer's W-storage.

```
00        DIRECTORYΔΔΔ    }   Header
01        ΔΔΔΔΔΔΔΔΔΔΔΔ    }
02        00W0000OWxxx
03        W-STORAGEΔΔΔ
04        ENDΔDIRECTRY
 .                        |
 :          Zero fill     |   Sentinels
 .                        |
59        ENDΔDIRECTRY
```

W000 - W-storage always starts with the zero word.

Wxxx - This is the highest number assigned to a W-storage word by the programmer.

In Sample Problem 2, only one word of W-storage is used.

Therefore, the Directory is:

```
00        DIRECTORYΔΔΔ      }  Header
01        ΔΔΔΔΔΔΔΔΔΔΔΔ      }
02        00W00000W000
03        W-STORAGEΔΔΔ
04        ENDΔDIRECTRY      )
:          Zero fill        )  Sentinels
59        ENDΔDIRECTRY      )
```

The problem has now been completely prepared for *FLOW-MATIC*.. The arrangement of the input tape is shown in Figure 33. Note that there are two additions to the format of the tape: the W-storage Data Design, and the Directory. The Directory must follow all Data Designs and precede the *FLOW-MATIC* code.

FLOW-MATIC INPUT TAPE

ABC MANUFACTURING COMPANY INVENTORY PROBLEM 2

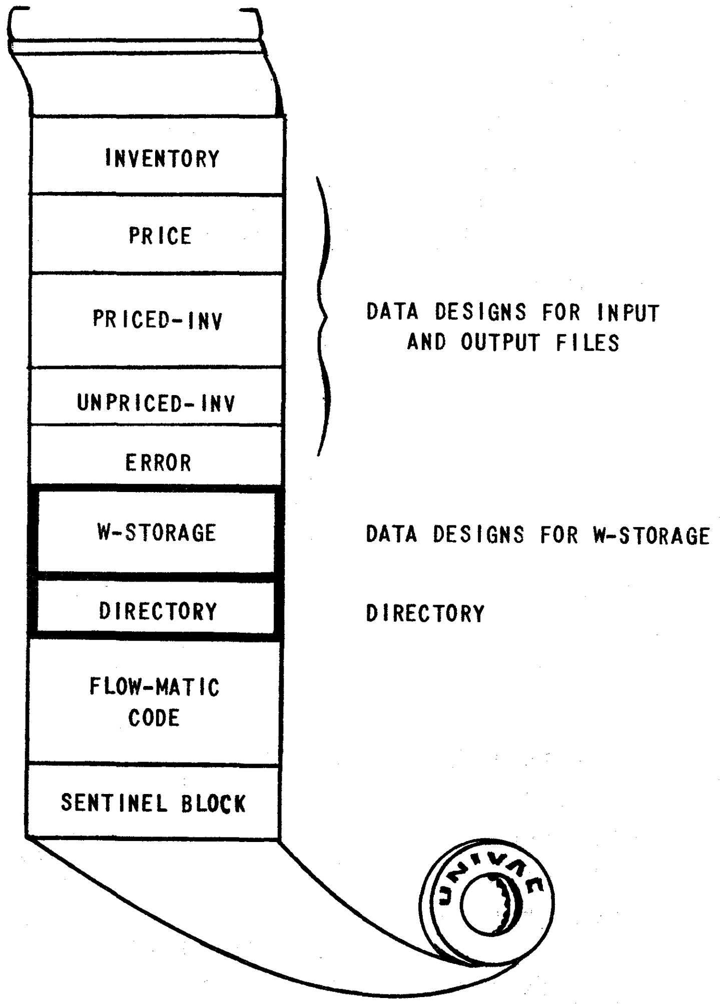


FIGURE 33

As implied above, it is sometimes useful to include some information under the Item section in the W-storage Data Design. For example, how can several entire items from a file be saved in W-storage? This is done by naming sub-items, groups of UNIVAC words, within W-storage and entering the assigned names of such sub-items in the Item design for W-storage. This use of sub-item names is exactly parallel to that discussed in Chapter 4, where the Data Designs for the input and output items are described.

Suppose it were required by this problem to store both the Inventory item and Price item. Then sub-items for each item in W-storage can be named and described as shown below:

```
00      NAMEΔOFΔFILE    }   Name of File
01      W-STORAGEΔΔΔ    }
02      ITEMΔDESIGNΔ    }   Header
03      ΔΔΔΔΔΔΔΔΔΔΔΔ    }
04      INV-ITEMΔΔΔΔ    }   Two word packet
05      000000000009    }   describing inventory item
06      PRICE-ITEMΔΔ    }   Two word packet
07      000010000011    }   describing price item
08      ENDΔFILEΔDES    }
 :       Zero fill      }   Ending Sentinels
59      ENDΔFILEΔDES    }
```

The corresponding Directory entry is:

```
00      DIRECTORYΔΔΔ    }   Header
01      ΔΔΔΔΔΔΔΔΔΔΔΔ    }
02      00W00000W011
03      W-STORAGEΔΔΔ
04      ENDΔDIRECTRY    }
 :       Zero fill      }   Sentinels
59      ENDΔDIRECTRY    }
```

Item names need not describe the entire W-storage area. Single field names may also be used. For example, addition of two field names to the W-storage words reserved above changes the Data Design and Directory as follows:

```
00     NAMEΔOFΔFILE   }  Name of File
01     W-STORAGEΔΔΔ   }
02     ITEMΔDESIGNΔ   }  Header
03     ΔΔΔΔΔΔΔΔΔΔΔΔ   }
04     INV-ITEMΔΔΔΔ   }  Two word packet
05     000000000009   }  for inventory item
06     PRICE-ITEMΔΔ   }  Two word packet
07     000010000011   }  for price item
08     FIELDΔDESIGN   }  Header
09     ΔΔΔΔΔΔΔΔΔΔΔΔ   }
10     FIELD-1ΔΔΔΔΔ
11     000012000000   }  4 word packet
12     000003///560   }  for 1st field
13     000011111100   }
14     FIELD-2ΔΔΔΔΔ
15     000013000000   }  4 word packet
16     000002///1CO   }  for 2nd field
17     000000000000   }
18     ENDΔFILEΔDES   }
 :       Zero fill    }  Ending Sentinels
59     ENDΔFILEΔDES   }
```

```
00     DIRECTORYΔΔΔ   }  Header
01     ΔΔΔΔΔΔΔΔΔΔΔΔ   }
02     00W00000W013
03     W-STORAGEΔΔΔ
04     ENDΔDIRECTRY   }
 :       Zero fill    }  Sentinels
59     ENDΔDIRECTRY   }
```

The programmer designs the W-storage in a manner similar to the design of data files themselves, following the same general rules. Following this, he writes a Directory stating the total W-storage required.

Whenever W-storage is utilized, the Directory Block must be written and it must immediately follow the blocks containing Data Designs and W-storage on the input tape. (Fig. 33)

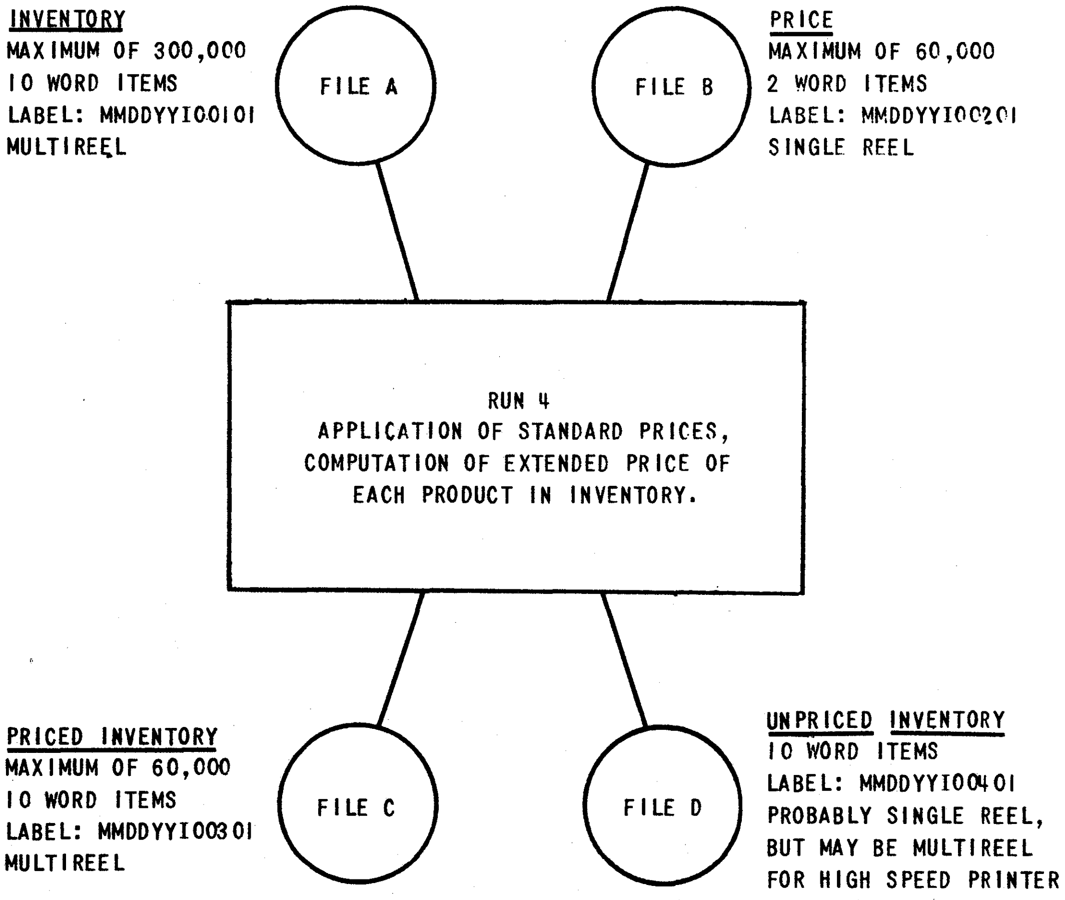
chapter 6

Relative Machine Coding, X-1

Suppose that a problem requires a function which does not appear in the current list of *FLOW-MATIC* functions. Consider the concrete example in Sample Problem 3. Suppose the definition of the original Sample Problem is altered once again, by the fact that duplicate product numbers in inventory items are not errors, but additional data to be processed. These items may be considered to originate in the different production plants of the ABC Manufacturing Company, and the problem is now to find the total on-hand balance for each product, to assign the appropriate unit price, and in addition to compute the total dollar value of these balances. This total dollar value will be referred to as the extended price for each product. Again, if there is no price available to apply to a given product number, an unpriced inventory item is to be created, but now this item will carry the total quantity on hand in all of the production plants.

The process chart for Sample Problem 3 looks like the original (compare Figures 34 on next page and 16 on page 19), but the logical statement of the problem is quite different.

ABC MANUFACTURING COMPANY INVENTORY PROBLEM 3
ALL FILES SEQUENCED BY PRODUCT NUMBER



Conventions

(1) LABELS IN WORD 03 OF FIRST BLOCK ON EACH REEL.

(2) BLOCK COUNTS IN WORD 01 OF LAST ITEM IN SENTINEL BLOCK.

(3) SENTINELS ARE ZZZZZZZZZZZY FOR END OF REEL AND ZZZZZZZZZZZZ FOR END OF FILE. THESE ARE LOCATED IN THE KEY WORD POSITION (WORD 000) OF FIRST INVALID ITEM AND LAST ITEM OF SENTINEL BLOCK.

FIGURE 34

Assume at the start that the first item of each of the input files is available.

I Compare the product number of the Inventory item with the product number of the Price item.

  a. If the Inventory product number is less, go to step II.
  b. If the product numbers are equal, go on to step III.
  c. If the Price product number is less, go on to step VIII.

II Prepare an Unpriced Inventory item and set step VII to second condition. Then go on to step IV.

III Prepare a Priced Inventory item and set step VII to first condition.

IV Store the product number and quantity from the Inventory file.

V Read the next Inventory item and go on to step VI. Or, if the Inventory file is exhausted, perform step VII, wind up the problem, and stop.

VI Compare the new product number with the stored product number.

  a. If the product numbers are equal, add the new quantity to the stored quantity and go back to step V.
  b. If the product numbers are not equal, go to step VII.

VII a. First condition. Insert the stored quantity (total on hand) in the Priced Inventory; compute extended price and insert it in the item; write a Priced Inventory item. Then go back to step I.

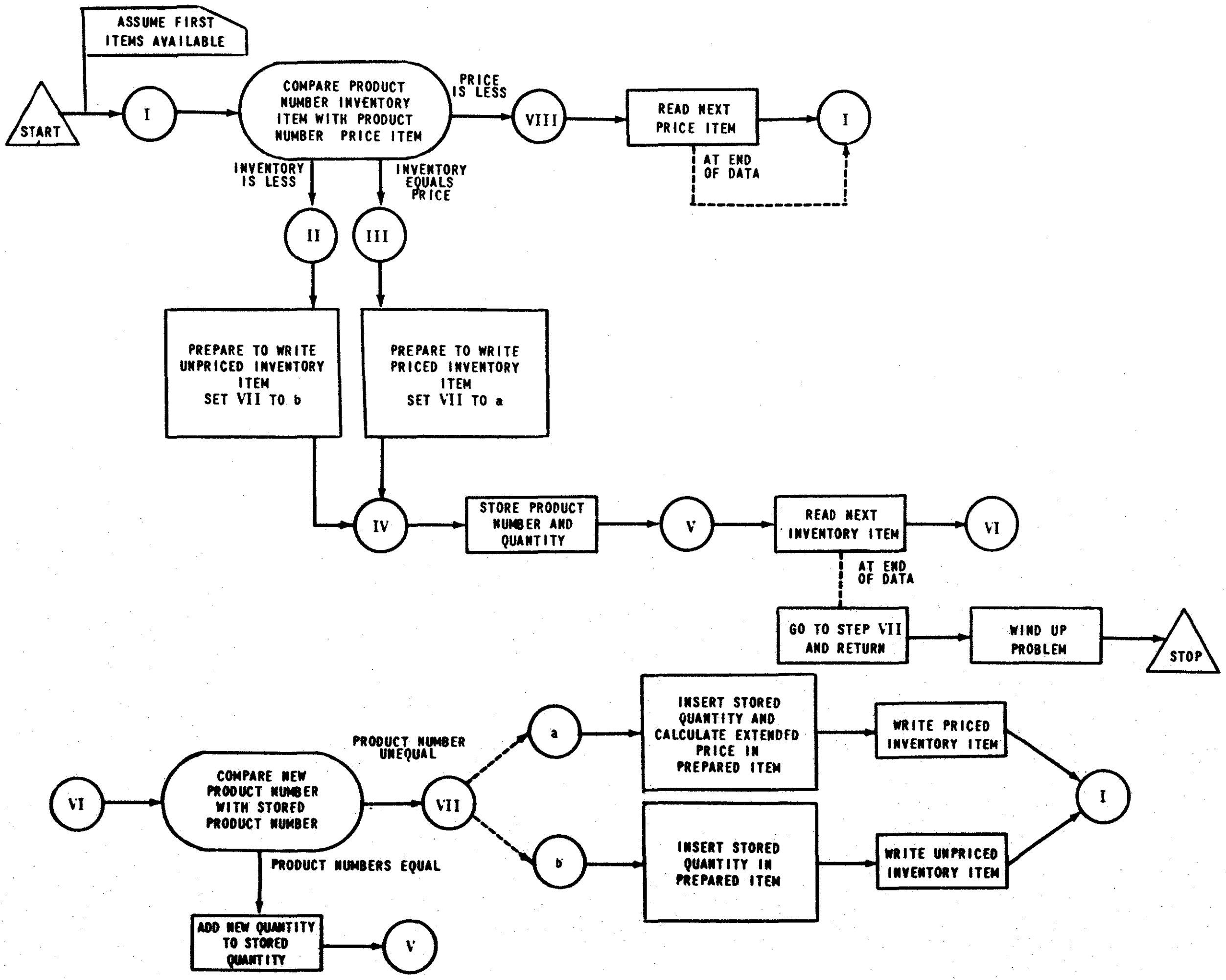
  b. Second condition. Insert the stored quantity in the Unpriced Inventory; write an Unpriced Inventory item. Then go back to step I.

VIII Read the next price item, and go back to step I.

The block chart for Sample Problem 3 is shown in Figure 35.

BLOCK CHART

ABC MANUFACTURING COMPANY INVENTORY PROBLEM 3



ASSUME FIRST ITEMS AVAILABLE
START
I
COMPARE PRODUCT NUMBER INVENTORY ITEM WITH PRODUCT NUMBER PRICE ITEM
PRICE IS LESS
VIII
READ NEXT PRICE ITEM
I
AT END OF DATA
INVENTORY IS LESS
INVENTORY EQUALS PRICE
II
III
PREPARE TO WRITE UNPRICED INVENTORY ITEM SET VII TO b
PREPARE TO WRITE PRICED INVENTORY ITEM SET VII TO a
IV
STORE PRODUCT NUMBER AND QUANTITY
V
READ NEXT INVENTORY ITEM
VI
AT END OF DATA
GO TO STEP VII AND RETURN
WIND UP PROBLEM
STOP
VI
COMPARE NEW PRODUCT NUMBER WITH STORED PRODUCT NUMBER
PRODUCT NUMBER UNEQUAL
VII
a
b
INSERT STORED QUANTITY AND CALCULATE EXTENDED PRICE IN PREPARED ITEM
INSERT STORED QUANTITY IN PREPARED ITEM
WRITE PRICED INVENTORY ITEM
WRITE UNPRICED INVENTORY ITEM
I
PRODUCT NUMBERS EQUAL
ADD NEW QUANTITY TO STORED QUANTITY
V

FIGURE 35

The *FLOW-MATIC* chart in Figure 36 shows two operations which require the use of new functions; Operation 11, addition, to find the total quantity on hand; and Operation 15, multiplication, to compute the extended price for the priced inventory file. The *FLOW-MATIC* system makes provision for functions which are not immediately available as a part of its library. These functions can be written by a UNIVAC programmer in a form called X-1. Whenever X-1 coding is to be used, an entry is made in the *FLOW-MATIC* code consisting of the appropriate operation number, the name "X-1", and an English statement denoting the function to be performed by the X-1 coding. Although the English statement is not processed by the compiler, it is useful in making the *FLOW-MATIC* program complete and understandable.

The *FLOW-MATIC* chart (Figure 36 on pages 76 and 77) and code (Figure 37 on page 78) for Sample Problem 3 are completed as before, maintaining the use of English to describe the logical steps. Note particularly operations 11 and 15 which call for X-1 coding.

FLOW-MATIC CHART

ABC MANUFACTURING COMPANY INVENTORY PROBLEM 3

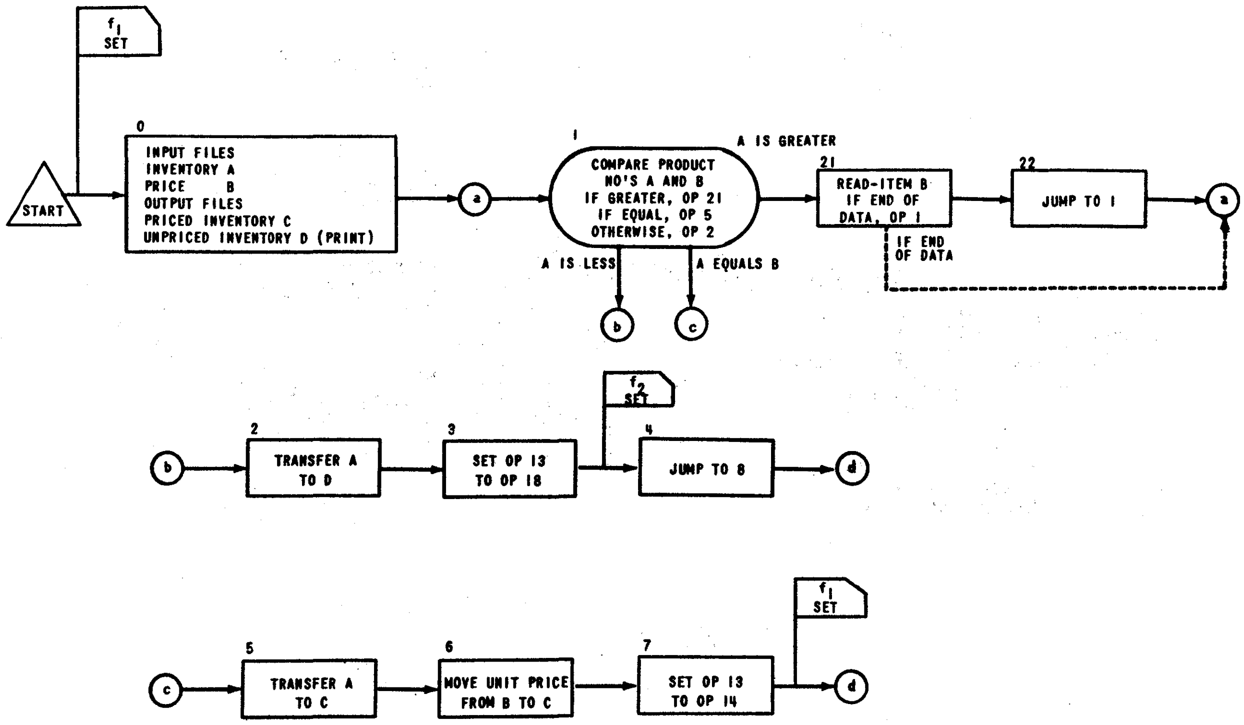


FIGURE 36

FLOW-MATIC CHART

ABC MANUFACTURING COMPANY INVENTORY PROBLEM 3

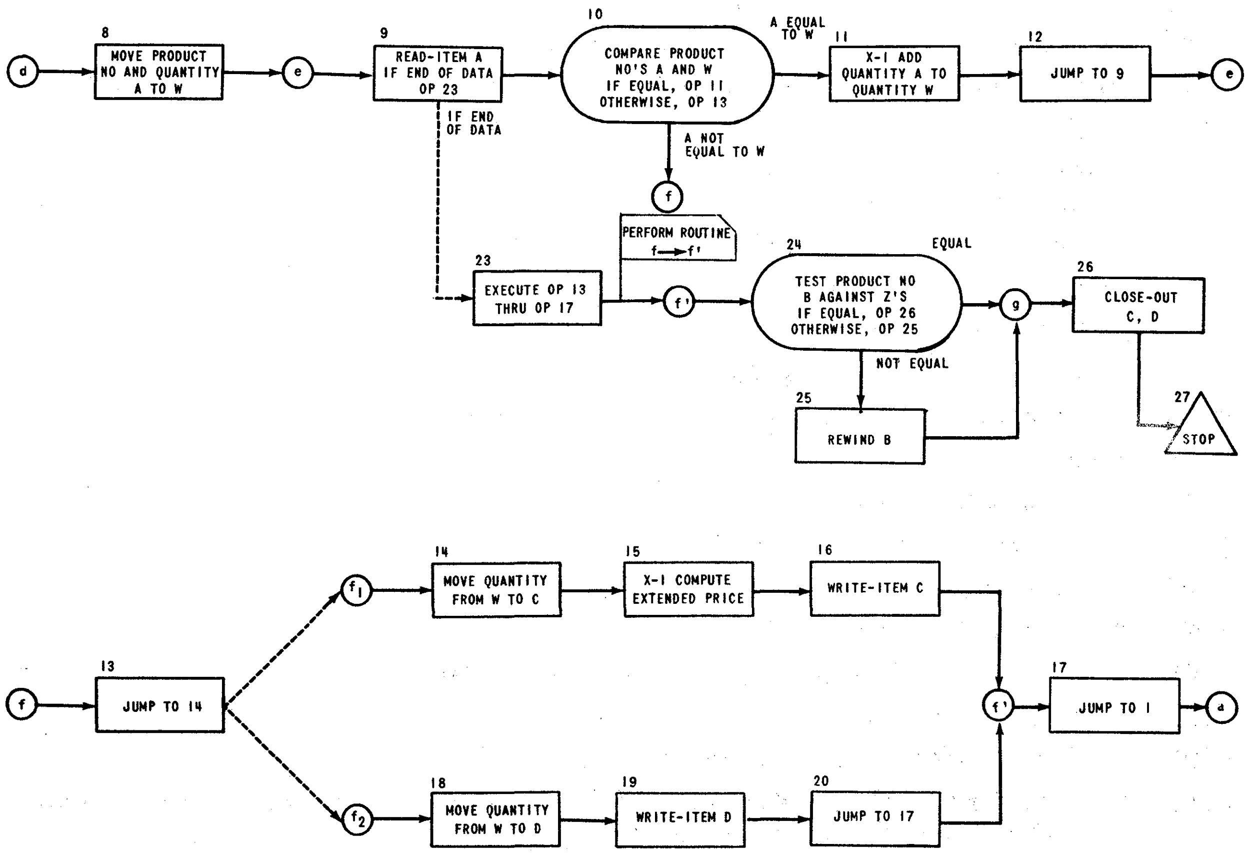

d
8
MOVE PRODUCT NO AND QUANTITY A TO W
e
9
READ-ITEM A IF END OF DATA OP 23
IF END OF DATA
10
COMPARE PRODUCT NO'S A AND W IF EQUAL, OP 11 OTHERWISE, OP 13
A EQUAL TO W
11
X-1 ADD QUANTITY A TO QUANTITY W
12
JUMP TO 9
e
A NOT EQUAL TO W
f
PERFORM ROUTINE f→f'
23
EXECUTE OP 13 THRU OP 17
f'
24
TEST PRODUCT NO B AGAINST Z'S IF EQUAL, OP 26 OTHERWISE, OP 25
EQUAL
g
26
CLOSE-OUT C, D
NOT EQUAL
25
REWIND B
27
STOP
f
13
JUMP TO 14
f1
14
MOVE QUANTITY FROM W TO C
15
X-1 COMPUTE EXTENDED PRICE
16
WRITE-ITEM C
f'
17
JUMP TO 1
a
f2
18
MOVE QUANTITY FROM W TO D
19
WRITE-ITEM D
20
JUMP TO 17

FIGURE 36
continued

FLOW-MATIC CODE

```
 (0)  INPUT INVENTORY FILE-A PRICE FILE-B : OUTPUT PRICED-INV FILE-C UNPRICED-
      INV FILE-D ; HSP D .
 (1)  COMPARE PRODUCT-NO (A) WITH PRODUCT-NO (B) ; IF GREATER GO TO OPERATION
      21 ; IF EQUAL GO TO OPERATION 5 : OTHERWISE GO TO OPERATION 2 .
 (2)  TRANSFER A TO D .
 (3)  SET OPERATION 13 TO GO TO OPERATION 18 .
 (4)  JUMP TO OPERATION 8 .
 (5)  TRANSFER A TO C .
 (6)  MOVE UNIT-PRICE (B) TO UNIT-PRICE (C) .
 (7)  SET OPERATION 13 TO GO TO OPERATION 14 .
 (8)  MOVE PRODUCT-NO (A) TO PRODUCT-NO (W) ; QUANTITY (A) TO QUANTITY (W) .
 (9)  READ-ITEM A ; IF END OF DATA GO TO OPERATION 23 .
(10)  COMPARE PRODUCT-NO(A) WITH PRODUCT-NO (W) : IF EQUAL GO TO OPERATION 11 ;
      OTHERWISE GO TO OPERATION 13 .
(11)  X-1 ADD QUANTITY (A) TO STORED QUANTITY (W) .
(12)  JUMP TO OPERATION 9 .
(13)  JUMP TO OPERATION 14 .
(14)  MOVE QUANTITY (W) TO QUANTITY (C) .
(15)  X-1 COMPUTE EXTENDED PRICE AND INSERT IN C ITEM .
(16)  WRITE-ITEM C .
(17)  JUMP TO OPERATION 1 .
(18)  MOVE QUANTITY (W) TO QUANTITY (D)
(19)  WRITE-ITEM D .
(20)  JUMP TO OPERATION 17 .
(21)  READ-ITEM B ; IF END OF DATA GO TO OPERATION 1 .
(22)  JUMP TO OPERATION 1 .
(23)  EXECUTE OPERATION 13 THROUGH OPERATION 17 .
(24)  TEST PRODUCT-NO (B) AGAINST ZZZZZZZZZZZZ : IF EQUAL GO TO OPERATION
      26 ; OTHERWISE GO TO OPERATION 25 .
(25)  REWIND B .
(26)  CLOSE-OUT FILES C , D .
(27)  STOP . (END)
```

ABC MANUFACTURING COMPANY INVENTORY PROBLEM 3

FIGURE 37

The W-storage design and Directory for this version of the problem are shown in Figure 38:

W-STORAGE DATA DESIGN

```
00    NAMEΔOFΔFILE   }  Name of file
01    W-STORAGEΔΔΔ   }
02    FIELDΔDESIGN   }  Header
03    ΔΔΔΔΔΔΔΔΔΔΔΔ   }
04    PRODUCT-NOΔΔ   |
05    000000000000   |  Four word packet
06    000002///1C0   |  describing product number
07    000000000000   |
08    QUANTITYΔΔΔΔ   |
09    000001000000   |  Four word packet
10    000003///760   |  describing quantity
11    000000111111   |
12    ENDΔFILEΔDES   |
.                    |
.      Zero fill     |  Ending sentinels
.                    |
59    ENDΔFILEΔDES   |
```

DIRECTORY

```
00    DIRECTORYΔΔΔ   }  Header
01    ΔΔΔΔΔΔΔΔΔΔΔΔ   }
02    00W00000W001
03    W-STORAGEΔΔΔ
04    ENDΔDIRECTRY   |
.                    |
.      Zero fill     |  Ending sentinels
.                    |
59    ENDΔDIRECTRY   |
```

ABC MANUFACTURING COMPANY INVENTORY PROBLEM 3

FIGURE 38

In addition to the above input to *FLOW-MATIC*, the necessary X-1 sections of relative machine coding are prepared and supplied to the compiler. This is a job for a person trained in UNIVAC coding since it entails the use of machine instructions. It does not, however, involve the use of actual machine addresses; rather, a system of relative and symbolic addresses, associated with the prepared item designs and field descriptions, is utilized. *FLOW-MATIC* accepts these X-1 sections as input, includes them in the sequence of operations as directed by the attached operation number, and converts the coder's symbolic and relative addresses to actual machine addresses.

In order to write the required X-1 coding the programmer needs the following information from the writer of the program:

(1) the operation number of the X-1 section which is to be written,

(2) the description of the function to be performed.

(3) the Data Designs and file letters of the files or items to be operated upon.

The *FLOW-MATIC* code, together with the Data Designs, is often sufficient information to describe the function to be coded, if the descriptive English in the X-1 operations is complete.

Detailed instructions for the use of the UNIVAC programmer in writing X-1 sections are included in Appendix C.

The prepared X-1 sections are Unityped on the input tape following the *FLOW-MATIC* code. The necessary sections for this Sample Problem are shown in Figure 39:

RELATIVE MACHINE CODING

```
00    X-1ΔΔΔΔΔΔ011    }  Header
01    BOWOO1A-A001    }
02    COWOO1 ~        }  Body of Coding
03    ENDΔSUBROUTN
 .                    }
 .                    }  Sentinels
 .                    }
09    ENDΔSUBROUTN
10    X-1ΔΔΔΔΔΔ015    }  Header
11    LOCOO1POC002    }
12    JOCOO3 ~        }  Body of Coding
13    ENDΔSUBROUTN
 .                    }
 .                    }  Sentinels
 .                    }
19    ENDΔSUBROUTN
 .
 .      Zero fill     }
 .      remainder     }  Fill
 .      of block      }
 .
59
```

ABC MANUFACTURING COMPANY INVENTORY PROBLEM 3

FIGURE 39

The input tape for Sample Problem 3 in Figure 40 shows the use of all present options in the *FLOW-MATIC* System. To summarize, a *FLOW-MATIC* input tape may contain:

*(1) Data Designs for the input and output files. These may be stored on the library tape for the system.

*(2) Data Design for W-storage. This may or may not be required, depending on the problem.

*(3) The Directory. Required if W-storage is used.

(4) *FLOW-MATIC* code.

*(5) X-1 Code sections. Required if called for in *FLOW-MATIC* code.

(6) Ending sentinel block.

* Optional

In summary the option of including X-1 coded sections is provided for two purposes:

(1) To allow a UNIVAC programmer to code a specialized function which does not have wide enough application to warrant permanent inclusion as a *FLOW-MATIC* library function.

(2) To allow a UNIVAC programmer to code a needed function which does not yet exist as a *FLOW-MATIC* operation.

The list of *FLOW-MATIC* functions will grow as application of the compiler indicates additional useful functions. Experience indicates that a large proportion of data-processing programs can be written by combining existing *FLOW-MATIC* functions. Specialization of a function to meet a particular requirement is made by the proper choice of option in the *FLOW-MATIC* code, and by the Data Designs. A major feature of the *FLOW-MATIC* Library is that it contains routines capable of generating machine coding to handle a large variety of programming and data situations. Thus a single entry in the library can produce a multitude of different, specialized codes which would be prohibitive to write by hand and to store on tape.

FLOW-MATIC INPUT TAPE
ABC MANUFACTURING COMPANY INVENTORY PROBLEM 3

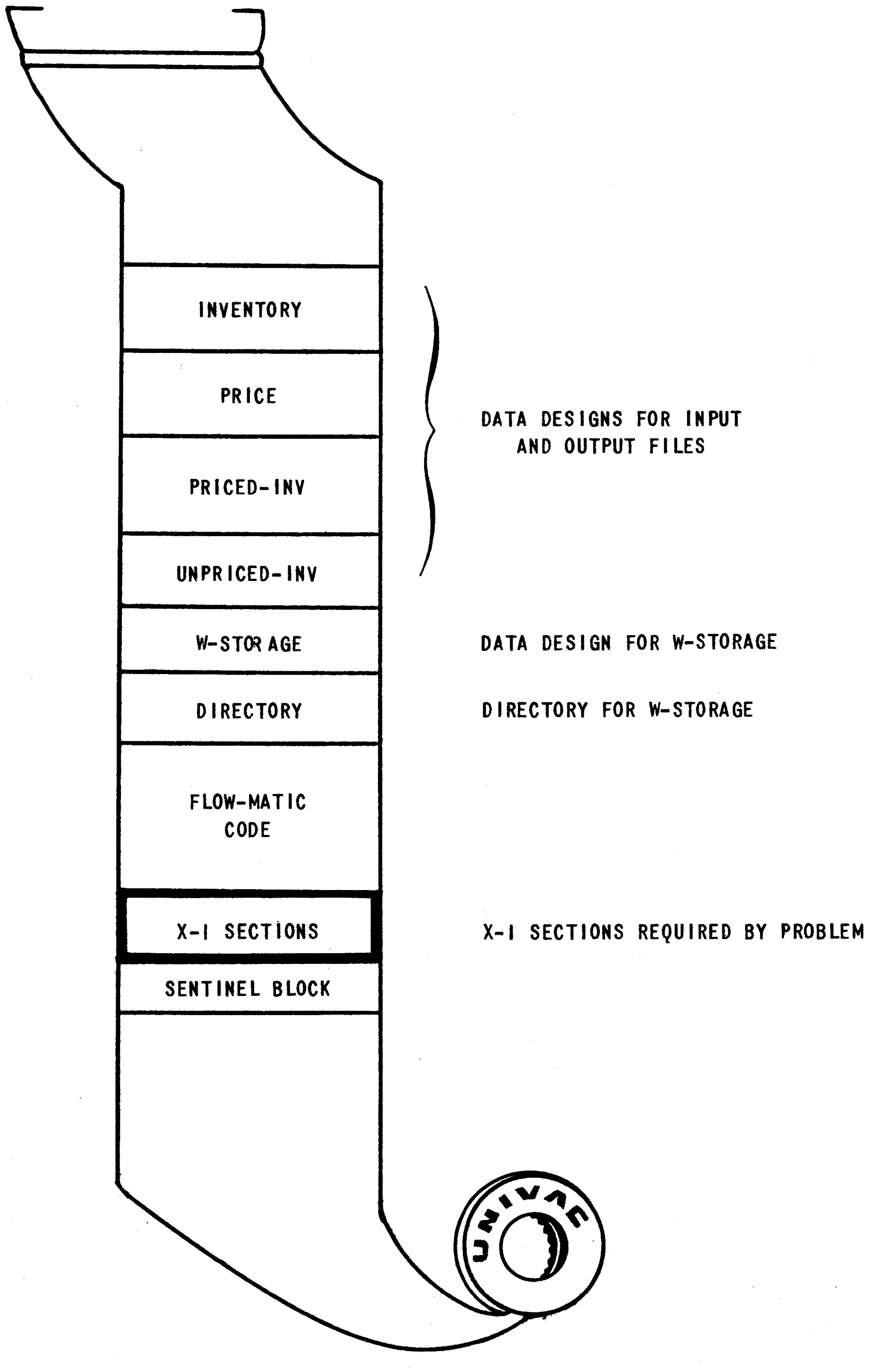


FIGURE 40

chapter 7

The Flow-Matic Compiling Routine

The process of preparing a problem for *FLOW-MATIC* has been described, using three different versions of the same basic problem. Now, what happens when the prepared input tape is used with *FLOW-MATIC* on the computer?

Figure 41 is a diagram of the four major phases of the compilation:

- Translation
- Selection
- Allocation
- Processing

FLOW CHART OF FLOW-MATIC PHASES

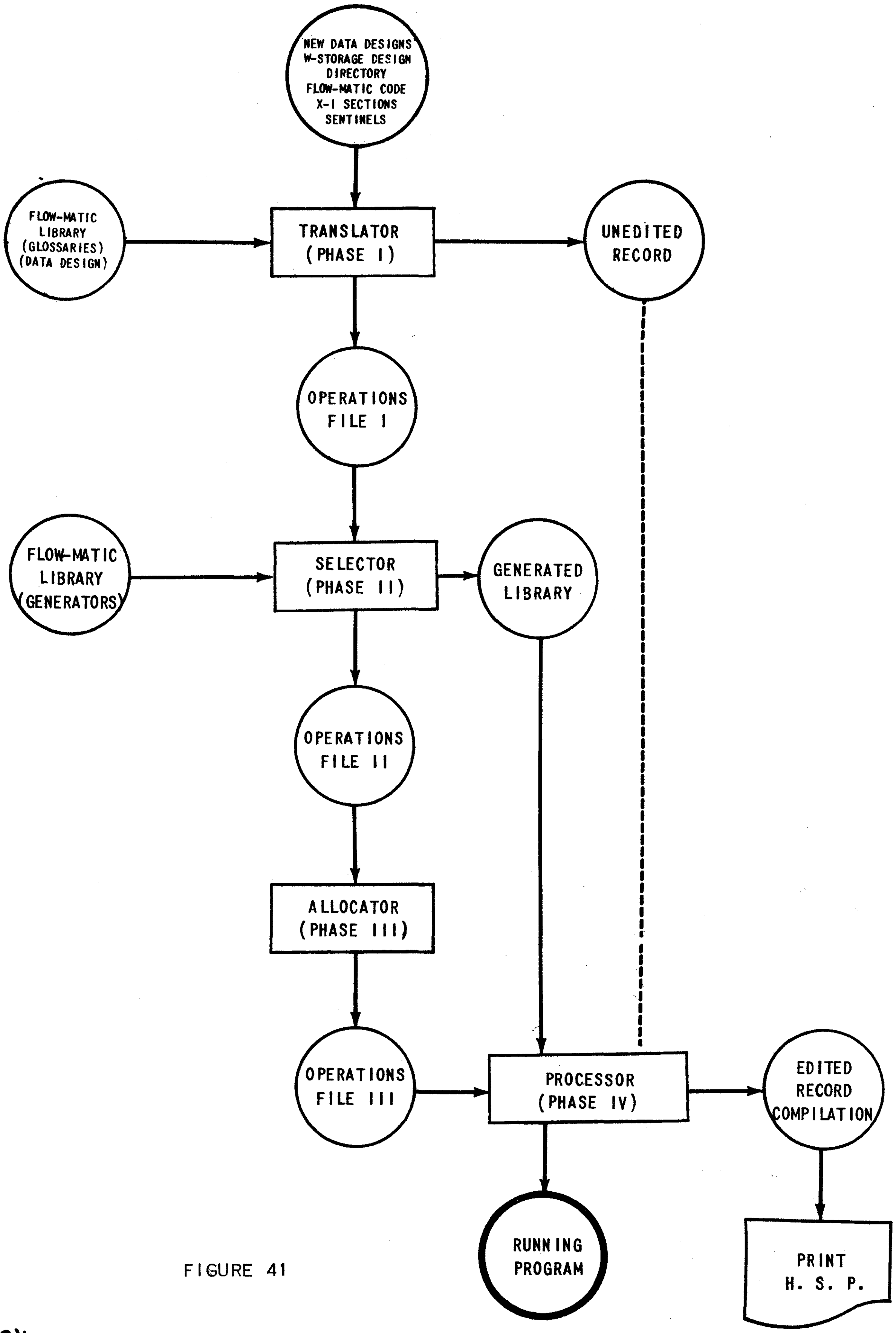


FIGURE 41

Translation: In this phase the *FLOW-MATIC* code is digested and condensed by the appropriate Glossary. Pertinent information from the stored and/or new Data Designs is added, creating the information listed in a standard format, called Operations File 1. The Translator also begins a list known as the Unedited Record, used in the final phase.

Selection: The purpose of the Selector phase is to choose from the *FLOW-MATIC* library the required functions and to produce for each operation the specialized coding to handle the fields, items, or files mentioned in the *FLOW-MATIC* code. These pieces of coding are also arranged in a standard format, and the entire set is called the Generated Library. In addition, during the Selector phase, supplementary information is added to Operations File 1, producing Operations File 2.

Allocation: The Allocator phase works only on Operations File 2, assigning data storage areas in the Memory as required by the program, and assigning a fixed Memory address for each piece of coding listed. These fixed Memory addresses are inserted in Operations File 2, producing Operations File 3.

Processing: The final phase combines Operations File 3 and the Generated Library, producing a program tape in machine code. Since all assigned locations are listed in Operations File 3, and all pieces of coding are contained in the Generated Library, the Processor simply inserts proper machine addresses in the generated pieces of coding and assembles them in proper order. In the Processing phase an Edited Record of the compilation is created from the Unedited Record for use both as a printed record and as an aid in debugging, if necessary.

During the Processing phase, a series of printouts is given. This list gives the general layout of the program, and is designed for a programmer's use in making a Codedit or Analyzer of the program tape.

In order to compile the first Sample Problem described in this manual, the appropriate tapes are mounted on the proper Uniservos, and the normal UNIVAC starting procedure is followed. (See Appendix D for detailed operating instructions.)

The normal printouts which occur during the compilation of the first sample problem are:

```
                         Printout                                            Explanation

FILE  00A000000000 SERVOS 333333333333 444444444444               Servo allocations
FILE  00B000000000 SERVOS 555555555555                            for input and out-
FILE  00C000000000 SERVOS 666666666666 777777777777               put files
FILE  00D000000000 SERVOS 111111111111 222222222222
END TRANS                                                         End of Translation
END SELECTOR                                                      End of Selection
END ALLOCTR                                                       End of Allocation
COMPILED PROGRAM                    STARTS   BLKS
INITIAL BLOCK                        0000     01
READS FOR SEGMENT 001                0940     01                  Description of com-
CODING FOR SEGMENT 001               0000     06                  piled program tape
PROGRAM ON SERVO 4   008 BLOCKS
                                                                  End of Processor,
END PROC I                                                        part I
TYPE IN PG. HEADER INFO                                           A request for three
                                                                  type-ins
                                                                     1. One word for
                                                                        name of run
                                                                     2. One word for
                                                                        programmer's
                                                                        name
                                                                     3. One word for
                                                                        date
EDITED RECORD ON SERVO 5   119 BLOCKS                             End of compilation
```

The printouts giving a description of the compiled program tape show that not all of the blocks are arranged to fall into consecutive storage locations. The compiled program has been designed so that the coding which reads in the program falls into the highest block address available. This block is later filled with data. A diagram of the compiled program tape is shown in Figure 42.

DIAGRAM OF COMPILED FLOW-MATIC PROGRAM TAPE

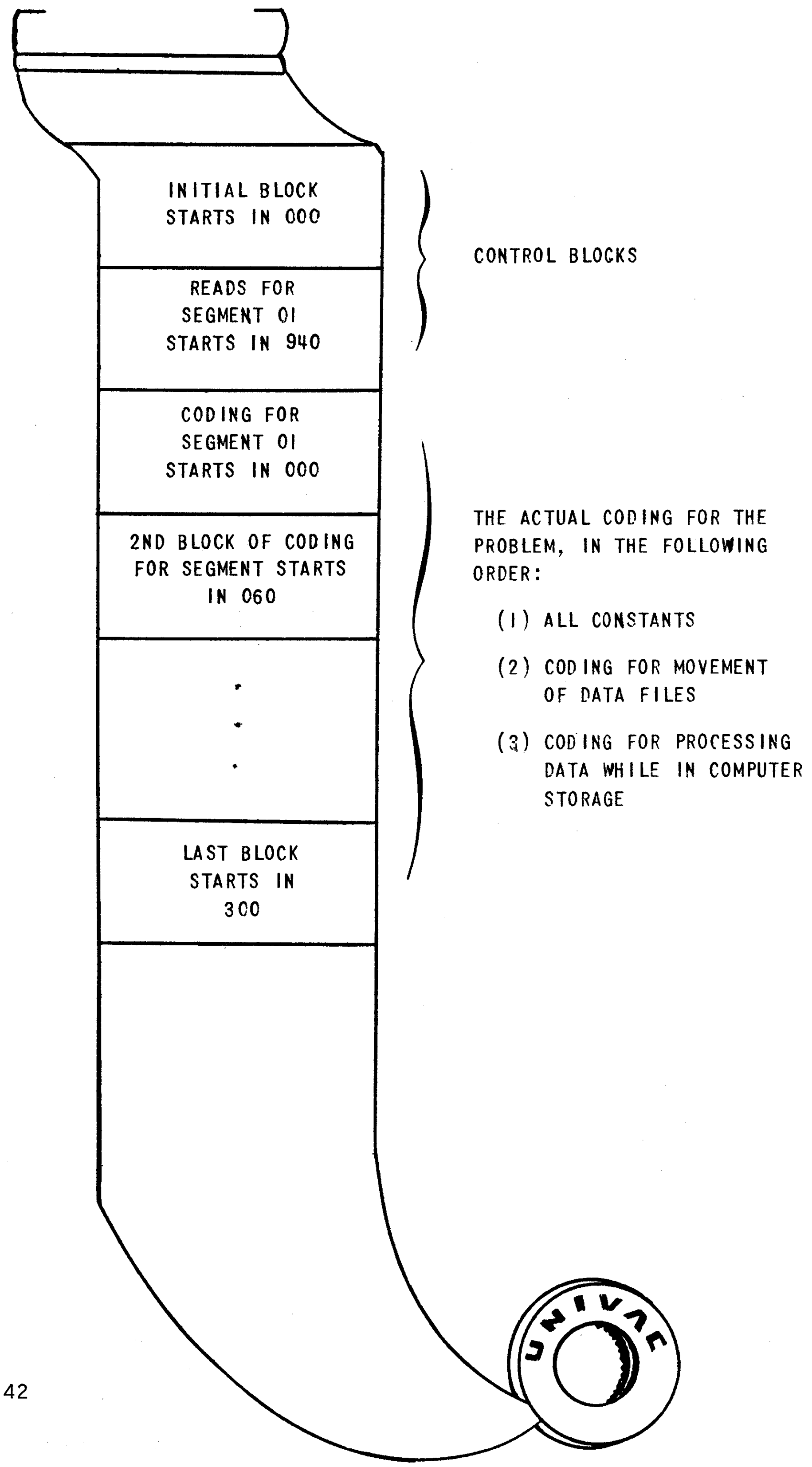

INITIAL BLOCK STARTS IN 000
READS FOR SEGMENT 01 STARTS IN 940
CONTROL BLOCKS
CODING FOR SEGMENT 01 STARTS IN 000
2ND BLOCK OF CODING FOR SEGMENT STARTS IN 060
LAST BLOCK STARTS IN 300
THE ACTUAL CODING FOR THE PROBLEM, IN THE FOLLOWING ORDER:
(1) ALL CONSTANTS
(2) CODING FOR MOVEMENT OF DATA FILES
(3) CODING FOR PROCESSING DATA WHILE IN COMPUTER STORAGE
UNIVAC

FIGURE 42

The Edited Record of compilation, a sample of which is shown in Figure 43, contains the following:

(1) A listing of the *FLOW-MATIC* input tape for the problem, with the exception of the Directory. It may contain Data Designs, W-Storage, *FLOW-MATIC* Code, X-I Sections.

(2) A table showing allocation of data storage areas with their related symbolic addresses.

(3) A list of the field names referred to in the *FLOW-MATIC* Code together with their assigned addresses, both symbolic and actual.

(4) A description of the compiled program which contains for each operation number and function:

(a) The assigned starting and ending lines,

(b) The addresses of all constants used by the operation,

(c) The addresses of all exits from, and entrances to, the operation.

The compiled program tape can be tested by removing it from servo 4 where it has been written, and mounting it on the instruction tape servo. Data files and blanks are mounted on the other servos as required by the run.

When Sample Problem 1 is run, the following normal print-outs occur:

Printout	*Explanation*	*Action*
MMDDYY100101	This input tape has passed the label check.	None. Type-out for log purposes.
MMDDYY100201	This input tape has passed the label check.	None. Type-out for log purposes.
B.S.D. TAPE 111111111111 222222222222	*FLOW-MATIC* code called for High-Speed Printer output on servo 1 and alternate servo 2.	Depress Block Subdivide buttons 1 and 2. Hit start bar.
MT NXT RL S 333333333333	A reel of input has been processed.	Mount the next reel of this file on servo 3.
MMDDYY100102	This input tape has passed the label check.	None. Type-out for log purposes.
666666666666 MMDDYY100301 000000000073	A reel of output has been completed.	Remove and label output tape, servo 6, with label and block count printed.
111111111111 MMDDYY100401 000000000025	A reel of output has been completed.	Remove and label output tape, servo 1, with label and block count printed.

```
Op No.      Call Word        Start      End                        Line

                                                RETURN JUMPS
                                                OP 004             0306

 005        TRANSFER          0307      0307    JUMPS
                                                OP 006             0308

 006        MOVE              0308      0309    JUMPS
                                                OP 007             0310

                                                CONSTANTS
                                                  K504             0035

 007        WRITE-ITEM        0310      0310    JUMPS
                                                OP C03             0156
                                                OP 008             0311

                                                RETURN JUMPS
                                                OP C03             0164

 008        READ-ITEM         0311      0311    JUMPS
                                                OP C09             0312
                                                OP A13             0049

                                                RETURN JUMPS
                                                OP A13             0061

                                                CONSTANTS
                                                K503               0036

 009        JUMP              0312      0312    JUMPS
                                                OP C01             0301

 010        READ-ITEM         0313      0313    JUMPS
                                                OP 011             0314
                                                OP B13             0115

                                                RETURN JUMPS
                                                OP B13             0127

                                                CONSTANTS
                                                  K502             0037

 011        JUMP              0314      0314    JUMPS
                                                OP 001             0301

 012        SET               0315      0315    JUMPS
                                                OP 002             0304
                                                OP 013             0316

                                                RETURN JUMPS
                                                OP 009             0312
                                                OP 009             0312
```

SAMPLE EDITED RECORD LISTING

FIGURE 43

This sample shows the kind of listing produced for each problem but is not an actual listing from the Sample Problems given in this manual.

Although the analyst originally prepared the *FLOW-MATIC* code he more than likely is unfamiliar with UNIVAC machine code. If detailed inspection of the program should be necessary, the Edited Record has proved entirely adeqûate in providing information about the program to the trained UNIVAC programmer. This allows him to assist the analyst in debugging (checking).

Appendix A

Flow-Matic Functions,
Guide For Writing Flow-Matic Code,
Flow-Matic Statements

FLOW-MATIC FUNCTIONS

For Sample Problems 1, 2, 3

(This is not a complete list of available FLOW-MATIC Functions.)

CLOSE-OUT:	Terminates the output files and rewinds the output tapes.
COMPARE:	Examines two fields for magnitude and/or equality: branches accordingly.
EXECUTE:	Performs designated operation or sequence of operations.
INPUT:	Identifies the input and output files to be used and supplies the first item of each input file.
JUMP:	Alters the normal sequence of operations and follows the directed path.
MOVE:	Places one or many fields of data in any other fields.
READ-ITEM:	Supplies the next item of an input file. When there is no more data, terminates the file and takes the directed path within the problem.
REWIND:	Rewinds current reel of an input file.
SET:	Alters an operation, changing the order of execution.
STOP:	Rewinds the instruction tape and terminates the problem.
TEST:	Examines the field and a constant for magnitude and/or equality; branches accordingly.
TRANSFER:	Places one item or group of words in any other item or group of words of *equal* size.
WRITE-ITEM:	Sends an output item to the output file.

GUIDE FOR WRITING FLOW-MATIC CODE

1. A *FLOW-MATIC* code word contains a maximum of twelve digits, none of which is a space (Δ).

2. *FLOW-MATIC* code words are separated by spaces (Δ's).

3. Each statement (operation) contains up to a maximum of sixty *FLOW-MATIC* code words:

   a) excluding the operation number and ending period,

   b) including all other words and punctuation marks.

4. Punctuation is according to proper English usage:

   a) punctuation marks count as words,

   b) only the ending period is of critical importance.

5. Assigned field names or file names may contain hyphens if it is desirable to combine more than one English word into a single name.

6. Assigned field names are always followed immediately by the pertinent file letter enclosed in parentheses:

   a) in all other cases the file letter is not parenthesized.

7. All operation numbers are numeric.

8. The operation number sequence starts with zero:

   a) operation zero is always the input statement.

9. The operations are written in unbroken numeric sequence:

   a) there may be a maximum of 999 separate operations.

10. The last operation must be the stop operation followed by the word END in parentheses.

FORMAT FOR FLOW-MATIC CODE

1. It is recommended that each operation begin a new blockette. Each statement may be space-filled to the end of the given blockette.

2. The final block of *FLOW-MATIC* code must terminate with at least twelve full digits of spaces (Δ's).

3. The blocks (s) of *FLOW-MATIC* code and X-1 sections, if needed, are followed by a sentinel block containing Z's in words 000 and 059.

LEGEND FOR DESCRIPTIONS OF OPERATIONS

1. Lower case indicates information to be supplied by the programmer.

2. Brackets [ ] indicate options available to the programmer.

3. h = present operation number
   h_1, h_2, h_3... = other operation numbers.

4. f_1, f_2, f_3... = file letters assigned by programmer.

5. s_1, s_2, s_3... = servo numbers assigned by programmer.

6. field-name = name assigned by programmer to data field. e.g., STOCK-NUMBER

7 file-name = name assigned by programmer to data file. e.g., INVENTORY

FLOW-MATIC LIBRARY ROUTINES

*(FLOW-MATIC CODE FORMAT)**

CLOSE-OUT (h)ΔCLOSE-OUTΔ$\begin{bmatrix}\text{FILE}\\ \text{FILES}\end{bmatrix}$ Δf_1Δ[f_2Δf_3Δ........Δf_nΔ].

COMPARE

Option 1
(h)ΔCOMPAREΔfield-nameΔ(f_1)ΔWITHΔfield-nameΔ(f_2)Δ
;ΔIFΔEQUALΔGOΔTOΔOPERATIONΔh_1Δ
;ΔOTHERWISEΔGOΔTOΔOPERATIONΔh_2Δ.

Option 2
(h)ΔCOMPAREΔfield-nameΔ(f_1)ΔWITHΔfield-nameΔ(f_2)Δ
;ΔIFΔGREATERΔGOΔTOΔOPERATIONΔh_1Δ
;ΔOTHERWISEΔGOΔTOΔOPERATIONΔh_2Δ.

Option 3
(h)ΔCOMPAREΔfield-nameΔ(f_1)ΔWITHΔfield-nameΔ(f_2)Δ
;ΔIFΔEQUALΔGOΔTOΔOPERATIONΔh_1Δ
;ΔIFΔGREATERΔGOΔTOΔOPERATIONΔh_2Δ
;ΔOTHERWISEΔGOΔTOΔOPERATIONΔh_3Δ.

Option 4
(h)ΔCOMPAREΔfield-nameΔ(f_1)ΔWITHΔfield-nameΔ(f_2)Δ
;ΔIFΔGREATERΔGOΔTOΔOPERATIONΔh_1Δ
;ΔIFΔEQUALΔGOΔTOΔOPERATIONΔh_2Δ
;ΔOTHERWISEΔGOΔTOΔOPERATIONΔh_3Δ.

EXECUTE (h)ΔEXECUTEΔOPERATIONΔh_1Δ[THROUGHΔOPERATIONΔh_2Δ].

* **Although the FLOW-MATIC code statements for the Sample Problems do not show explicitly the spaces (Δ's) between words, they are included here for completeness and accuracy.**

INPUT

```
(O)ΔINPUTΔname-of-fileΔFILE-f1Δ [SERVOΔs1Δ         ]
                                [SERVOSΔs1Δ,Δs2Δ   ]
          name-of-fileΔFILE-f2Δ [SERVOΔs1Δ         ]
                    :           [SERVOSΔs1Δ,Δs2Δ   ]
  ;ΔOUTPUTΔname-of-fileΔFILE-f3Δ [SERVOΔs1Δ        ]
                                 [SERVOSΔs1Δ,Δs2Δ  ]
          name-of-fileΔFILE-f4Δ [SERVOΔs1Δ         ]
                    :           [SERVOSΔs1Δ,Δs2Δ   ]
[;ΔPRESELECTIONΔ]
[;ΔHSPΔf1Δ,Δf2Δ,Δ...Δfn Δ]
[;ΔT/CΔf1Δ,Δf2Δ,Δ...Δfn Δ]
            [ ON  ]
[;RERUNΔ    [WITH ]ΔOUTPUTΔf1Δ].
            [FROM ]
```

SPECIAL NOTES

1. The assigned file-name may not begin with the digits FILE-.

2. If servo numbers are not specified, the compiler will assign them, reserving the proper servos for tapes for HSP or T/C, as stated.

3. For a single input file, continuous reads will be provided. For two or three way input, standby coding will be provided, unless preselection is specified. Up to eight way input coding will be provided in the preselection option, and up to five full-word keys may be used.

4. The normal and error print-outs which may occur in the execution of this coding are self-explanatory. In addition, breakpoint 1 is used throughout the coding to provide operating options. For example, if a tape label fails the check, an error print-out gives this information to the operator; he may then force transfer on breakpoint 1 to proceed, if desired. In all cases where it is desired to bypass the error, the action is to force transfer on breakpoint 1.

5. If it is desired to begin the problem over, the operating instructions are:

   a - Rewind all tapes except the instruction tape.
   b - Clear C and rI.
   c - No transfer on Breakpoint 2.

   This coding is always provided.

6. If rerun coding has been requested, the operating instructions are:

   a - Rewind all tapes except the instruction tape.
   b - Clear C and rI.
   c - Force transfer on Breakpoint 2.

   The problem will be resumed from the last completed output reel, as specified in the input statement.

JUMP (h)ΔJUMPΔTOΔOPERATIONΔh$_1$Δ.

MOVE (h)ΔMOVEΔfield-nameΔ(f$_1$)ΔTOΔfield-nameΔ(f$_2$)Δ
[,Δfield-nameΔ(f$_3$)Δ
.
.
.
,Δfield-nameΔ(f$_n$)Δ]

[;Δfield-nameΔ(f$_1$')ΔTOΔfield-nameΔ(f$_2$')Δ
[,Δfield-nameΔ(f$_3$')Δ
.
.
.
,Δfield-nameΔ(f$_n$')Δ].

READ-ITEM
(h)ΔREAD-ITEMΔf$_1$Δ[;ΔIFΔENDΔOFΔDATAΔGOΔTOΔOPERATIONΔh$_1$Δ].

SPECIAL NOTES

1. Each input file mentioned in the INPUT statement must have at least one READ-ITEM operation.

2. At least one READ-ITEM operation, for each input file, must include the optional phrase, IF END OF DATA......

3. If two or more READ-ITEM operations, for a single input file, include the IF END OF DATA.......option, the operation numbers, h$_1$, must be identical.

REWIND (h)ΔREWINDΔf$_1$Δ[,Δf$_2$Δ,f$_3$...,Δf$_n$Δ].

SET (h)ΔSETΔOPERATIONΔh$_1$ΔTOΔGOΔTOΔOPERATIONΔh$_2$Δ
[,ΔOPERATIONΔh$_3$ΔTOΔGOΔTOΔOPERATIONΔh$_4$Δ
.
.
.
,ΔOPERATIONΔh$_5$ΔTOΔGOΔTOΔOPERATIONΔh$_6$Δ].

STOP (h)ΔSTOPΔ.Δ(END)

SPECIAL NOTES

1. The stop operation must be the highest numbered operation in the problem, and it must be followed by the word END in parentheses.

TEST

Option 1

(h)ΔTESTΔfield-nameΔ(f_1)ΔAGAINSTΔtest-valueΔ
;ΔIFΔGREATERΔGOΔTOΔOPERATIONΔh_1Δ
;ΔIFΔEQUALΔGOΔTOΔOPERATIONΔh_2Δ
[;ΔAGAINSTΔ...Δ]
;ΔOTHERWISEΔGOΔTOΔOPERATIONΔh_3Δ.

Option 2

(h)ΔTESTΔfield-nameΔ(f_1)ΔAGAINSTΔtest-valueΔ
;ΔIFΔGREATERΔGOΔTOΔOPERATIONΔh_1Δ
;ΔIFΔLESSΔGOΔTOΔOPERATIONΔh_2Δ
[;ΔAGAINSTΔ...Δ]
;ΔOTHERWISEΔGOΔTOΔOPERATIONΔh_3Δ.

Option 3

(h)ΔTESTΔfield-nameΔ(f_1)ΔAGAINSTΔtest-valueΔ
;ΔIFΔUNEQUALΔGOΔTOΔOPERATIONΔh_1Δ
[;ΔAGAINSTΔ...Δ]
;ΔOTHERWISEΔGOΔTOΔOPERATIONΔh_2Δ.

SPECIAL NOTES

1. The conditional phrases, IF GREATER, IF EQUAL, IF LESS, shown in Option 1 and 2 may appear singly or in any order in combinations of two. The phrase OTHERWISE must always appear, and must be written last.

2. If it is desired to use a test value of spaces or periods, the words SPACE, PERIOD, SPACES, or PERIODS should be used instead of the actual digits. In these cases, only one test value is acceptable. In all other cases, tests against many test values may be made in one operation.

TRANSFER

Option 1 (h)ΔTRANSFERΔf_1ΔTOΔf_2Δ.

Option 2 (h)ΔTRANSFERΔsub-item-nameΔINΔf_1ΔTOΔf_2Δ.

Option 3 (h)ΔTRANSFERΔf_1ΔT$\underline{O}$Δsub-item-nameΔINΔf_2Δ.

Option 4 (h)ΔTRANSFERΔsub-item-nameΔINΔf_1ΔT$\underline{O}$Δsub-item-nameΔINΔf_2Δ.

SPECIAL NOTES

1. Item or sub-item sizes in f_1 and f_2 must be equal in size.

WRITE-ITEM (h)ΔWRITE-ITEMΔf_1Δ.

Appendix B

Data Design Pre-Printed Forms

ANALYST: ____________________ PAGE: ______

DATE: ____________________

N	A	M	E	Δ	O	F	Δ	F	I	L	E	
												English name of file.
F	I	L	E	Δ	D	E	S	I	G	N	Δ	
Δ	Δ	Δ	Δ	Δ	Δ	Δ	Δ	Δ	Δ	Δ	Δ	
L	A	B	E	L	Δ	Δ	Δ	Δ	Δ	Δ	Δ	
												1.) Label with reel counter. 2.) If label variable all Δ's.
L	O	C	Δ	O	F	Δ	L	A	B	E	L	
0	0	0	0	0	0	0	0	0				Word location of label in label block.
M	U	L	T	I	Δ	R	E	E	L	Δ	Δ	
0	0	0	0	0	0	0	0	0	0	0		1 = Yes; 0 = No.
B	L	K	Δ	C	T	Δ	I	N	D	Δ	Δ	
0	0	0	0	0	0	0	0	0	0	0		1 = Block count desired; 0 = No block count.
B	L	K	Δ	C	T	Δ	L	O	C	Δ	Δ	
0	0	0	0	0	0	0	0	0				Word location of block count in last item of sentinel block. [000 to (item size-1)]
E	N	D	Δ	R	E	E	L	Δ	S	E	N	
												End of reel sentinel. If single reel, all Δ's.
E	N	D	Δ	F	I	L	E	Δ	S	E	N	
												End of file sentinel.
L	O	C	Δ	I	N	Δ	F	I	R	S	T	
0	0	0	0	0	0	0	0	0				Word location of sentinel in first invalid item. [000 to (item size-1)]
L	O	C	Δ	I	N	Δ	L	A	S	T	Δ	
0	0	0	0	0	0	0	0	0				Word location of sentinel in last item of sentinel block. [000 to (item size-1)]
												Other entries may be added here, each consisting of a title word and an information word; e.g., PARTΔBLKΔSEN ZZZZZZZZZZZX
												UNITYPIST NOTE: After the last entry skip to the next page.
												FLOW-MATIC DATA DESIGN FORM 1
												Remington Rand Univac DIVISION OF SPERRY RAND CORPORATION

ANALYST: ______________________ PAGE: ____

DATE: ______________________

I	T	E	M	Δ	D	E	S	I	G	N	Δ	
Δ	Δ	Δ	Δ	Δ	Δ	Δ	Δ	Δ	Δ	Δ	Δ	
I	T	E	M	Δ	S	I	Z	E	Δ	Δ	Δ	
O	O	O	O	O	O	O	O	O				1,2,3,4,5,6,10,12,15,20,30,60.
N	O	Δ	O	F	Δ	K	E	Y	S	Δ	Δ	
O	O	O	O	O	O	O	O	O	O	O		0,1,2,....,9 = number of keys
K	E	Y	Δ	1	Δ	Δ	Δ	Δ	Δ	Δ	Δ	
												Name of field, if no key ΔΔΔΔΔΔΔΔΔΔΔΔ

Further key entries may be added here, each consisting of KEYΔnΔΔΔΔΔΔΔ followed by the name of the field.

If there are Sub-items to be described, the descriptions are entered following the last Key entry. Sub-items are described with two-word packets consisting of the name of the Sub-item followed by a word in the format 000SSS000EEE, where SSS is the first word and EEE is the last word of the Sub-item, relative to the entire item.

UNITYPIST
NOTE: After the last entry skip to the next page.

FLOW-MATIC DATA DESIGN FORM 2

Remington Rand Univac
DIVISION OF SPERRY RAND CORPORATION

ANALYST: ______________________ PAGE: ____

DATE: ______________________

F	I	E	L	D	△	D	E	S	I	G	N	
△	△	△	△	△	△	△	△	△	△	△	△	
												Name of field.
0	0	0				0	0	0	0	0	0	Word location in item. [000 to (item size-1)]
0	0	0	0	0							0	Field descriptor of form 00000TPPSLN0*
												Extractor; if full-word field, all 0's.
0	0	0				0	0	0	0	0	0	
0	0	0	0	0							0	
0	0	0				0	0	0	0	0	0	
0	0	0	0	0							0	
0	0	0				0	0	0	0	0	0	
0	0	0	0	0							0	
0	0	0				0	0	0	0	0	0	
0	0	0	0	0							0	
0	0	0				0	0	0	0	0	0	
0	0	0	0	0							0	
0	0	0				0	0	0	0	0	0	
0	0	0	0	0							0	

An unlimited number of fields may be described using the same four-word packet format.

* Explanation of field descriptor:

T	=	Type of field. 1 - alphabetic 2 - alpha-numeric 3 - numeric
PP	=	Position of decimal point in relation to a reference point immediately to the left of the left-most digit of the field. 00 - coincident with reference point ɨɨ - not applicable nL - n positions to the left of the reference point nR - n positions to the right of the reference point (n = 1,2,...,9,A,B,...,Z)
S	=	Digit position of the sign. 1,2,...,9,A,B, or C ɨ - not applicable
L	=	Digit position of the left-most digit of the field, excluding sign. 1,2,...,9,A,B, or C
N	=	Number of digits in the field, excluding sign. 1,2,...,9,A,B, or C (A = 10, B = 11, C = 12)

<u>NOTE:</u> Place the sentinel END△FILE△DES immediately following the last four-word packet <u>and</u> in word 059 of that block.

FLOW-MATIC DATA DESIGN FORM 3

Remington Rand Univac
DIVISION OF SPERRY RAND CORPORATION

S1-1501

Appendix C

Relative Machine Coding, X-1

X-1 sections of a *FLOW-MATIC* program are hand-tailored sections of coding. Actual machine instructions are used, but a system of relative and symbolic addressing is used in place of actual machine locations. These X-1 addresses are always a combination of an alphabetic character together with three numeric digits.

Use	*Alphabetic*	*Numeric*
To address a field within an input or output item, or the item itself. The alphabetic is the assigned file letter of the field or item; the numeric is the word location within the item.	A B C D E F G H I	000 through item size minus one.
To address a field or unit in W-storage. The alphabetic is always W; the numeric is the word location within W-storage.	W	000 through maximum assigned by programmer in Directory.
To address a line of coding within this particular section. The alphabetic, M, indicates that this relative address is to be modified by the addition of the starting line assigned by the compiler.	M	000 through maximum used by coder in this section.
To address a line of coding in another X-1 section. The operation number of the section referred to is also required, and must follow the J notation in the next line of coding as: - U0J011 000000QP.021 meaning "jump to line M011 of X-1 operation 21."	J	000 through number of last line of coding in X-1 section referred to.
To address a temporary storage location, that is, a location whose contents are not to be preserved for future operations.	T	000 through maximum assigned by coder.

The X-1 sections are written in blockette rather than block form, starting each new section at the beginning of a blockette and continuing for as many blockettes as required. They may be typed in any order of operation numbers. The format of an X-1 section is as follows:

Header - One word containing the *FLOW-MATIC* operation number for which the coding is written, as "X-1ΔΔΔΔΔΔ012." This header is not counted as an M address.

Body of coding - The coding is written starting with line number M000 and utilizing the appropriate address symbols given above. The extra lines ("000000QP.021") used with the J notation are not counted as M addresses.

Constants - Following the title "CONSTANTSΔΔΔ", numeric or alphabetic quantities are listed which are not to be modified by the compilation process. The title is not counted as an M address; but the actual constants are assigned M addresses in sequence with other lines of the section. These constants are stripped out by the compiler and included with other constants from the remainder of the program, and all duplicates are eliminated. Since in the final compiled program, constants are not listed in their original order, addressing a series of constants by incrementing an instruction is not permissible. If it is desired to use such a technique, and if the constants are numeric in the third and ninth digits, they may be included preceding the title as a part of the main body of coding. Any counters incremented during execution of the program must also be included in the main body of the coding, not in the Constants Section.

Code Constants - Under the title word "CODEΔCONSTANTSΔ" are listed those constants containing address symbols in the third and ninth digits which are to be converted by the Compiler. The general rules stated in the preceding section apply to this group also. All X-1 symbols may be used except the "J" and the "M"; where it is required to use these symbols, the words containing them must be carried in the body of the coding.

A maximum of 59 constants and code constants together may be written in any single X-1 section.

End Sentinel - the word "ENDΔSUBROUTN" is written following the last line of the section and in word 09 of the blockette (not necessarily in 59 of the block). A subsequent X-1 section begins (with the header) in word 00 of the next blockette, and any partial blocks are simply zero filled.

Since the constants and code constants are removed by the compiler, and placed in a constant pool for the entire program, the exit from an X-1 section is normally through the last line of coding to the next *FLOW-MATIC* operation. If it is desired that this exit be other than the last line of coding, the programmer uses the J notation to address line 000 of the next operation.

A non-functional section of X-1 coding, which illustrates all of the above comments, is shown in Figure 44 on page 109.

Normal Printouts

During compilation: None

Error Printouts

During compilation (Selector Phase)

Printout	*Meaning*	*Action*
CANNOTΔFINDΔ X-1ΔΔΔΔΔΔhhh SUBROUTNΔONΔ INPUTΔTAPEΔΔ	Operation hhh in *FLOW-MATIC* code calls for an X-1 section, but no section with this number appears on the input tape.	Type in correct header as X-1ΔΔΔΔΔΔhhh
BADΔADDRESSΔ X-1ΔΔΔΔΔΔhhh OOMMMMOOOOOO CCCCCCCCCCCC	Line MMMM of X-1 section hhh contains an alphabetic other than A-I, J, M, T, W, as shown in CCCCCCCCCCCC.	Type in corrected line of coding CCCCCCCCCCCC
WRONGΔRELΔAD X-1ΔΔΔΔΔΔhhh OOMMMMOOOOOO CCCCCCCCCCCC	Line MMMM of X-1 section hhh contains an M reference greater than the M address of the last line of the section.	Type in corrected line of coding, CCCCCCCCCCCC
CANNOTΔUSEΔΔ X-1ΔΔΔΔΔΔhhh OOMMMMOOOOOO CCCCCCCCCCCC	Line MMMM of X-1 section hhh is a code constant and contains a reference to a non-permissible M address.	Type in corrected line of coding, CCCCCCCCCCCC
ioJxxxioJyyy	The printout shows a line of coding which has at least one J address, but no line with op nos. follows it (where i equals any instruction.)	Type in op no. or op nos. as OP.hhh in proper half-word(s).
HALFΔWORDΔΔΔ ADDRESSΔINΔΔ X-1ΔΔΔΔΔΔhhh OOMMMMOOOOOO CCCCCCOOOOOO	Line MMMM of X-1 section hhh contains an invalid file letter as shown in instruction CCCCCC.	Type in correction CCCCCCOOOOOO.

SAMPLE X-I SECTION

Block Address			*Symbolic Address*
00	X-1ΔΔΔΔΔΔ012	Header	
01	BOA001LOM008		M000
02	—— QOM006		M001
03	A-M009COA001		M002
04	BOM000A-M010		M003
05	LOM011QOM006	Body of Coding	M004
06	HOM000UOJ000		M005
07	—— OP.013		
08	BOM012COM000		M006
09	—— UOM000		M007
10	CONSTANTSΔΔΔ	Title	
11	000000000001		M008
12	000006000006	Constants	M009
13	000001000000		M010
14	CODEΔCONSTSΔ	Title	
15	BOA010LOM008	Code Constants	M011
16	BOA000LOM008		M012
17	ENDΔSUBROUTN		
18	—— ——		
19	ENDΔSUBROUTN		

FIGURE 44

Appendix D

Flow-Matic Operating Instructions

FLOW-MATIC COMPILATION

The following information about the operation of a *FLOW-MATIC* compilation indicates the print-outs and action to be taken for a *normal* (i.e., error-free) compilation.

PART I. OPERATING INSTRUCTIONS FOR COMPILATION

Servo Allocation:

1	*FLOW-MATIC* Compiler	(ring)
2	*FLOW-MATIC* Library	(ring)
3-7	Blanks	
8	*FLOW-MATIC* Code	(ring)
9	Blank (Rerun)	

Initial Instructions:

a. Mount tapes as indicated above.
b. Supervisory Control Printer on normal.
c. No block subdivision.*
d. Initial read servo 1 (compiler) and hit start bar.

Breakpoints:

1	Force no transfer to omit use of servo 9 for Rerun.
5	Release after Block Subdividing servo 5.

* It is necessary to BSD servo 5 in order to allow the Edited Record to be printed. To achieve this, set breakpoint 5 at start of compilation and follow instructions in NORMAL PRINTOUTS.

PART II. NORMAL PRINT-OUTS DURING COMPILATION

Print-Out				*Action*	*Explanation*
FILE 00f000000000 SERVOS aaaaaaaaaaaa bbbbbbbbbbbb				None	aaaaaaaaaaaa = servo number for initial reel
FILE 00f000000000 SERVOS aaaaaaaaaaaa bbbbbbbbbbbb					bbbbbbbbbbbb = servo number for alternate reel
.					f = input or output file letter
.					
.					
END TRANS				None	
END SELECTOR				None	
END ALLOCTR				None	
COMPILED PROGRAM		STARTS	BLOCKS	None	
INITIAL BLOCK		0000	01		
READS FOR SEGMENT	001	0940	01		
RERUN		0940	01		
		0760	02		
CODING FOR SEGMENT	001	0000	nn		nn = number of blocks in 1st segment
CODING FOR SEGMENT	002	CCCC	mm		mm = number of blocks in 2nd segment, minus 1
.		DDDD	01 (see Cycling)		CCCC = location of 1st block in 2nd segment
.					DDDD = location of last block in 2nd segment
.					
PROGRAM ON SERVO 4		XXX BLOCKS			XXX = total number of blocks in compiled tape
END PROC 1.				None	
B.S.D. SERVO 5				Block subdivide servo 5, release breakpoint #5 and hit start bar	
TYPE IN PG. HEADER INFO				Type in three (3) words of run identification e.g., name of run, programmer, date	
EDITED RECORD ON SERVO 5		YYY BLOCKS		None	YYY = number of blocks in EDITED RECORD

CYCLING

Note that the last block of the second, third, ..., segments may have a special address, out of sequence with the other blocks in the segment. This occurs when the last block of a segment is a partial block. The compiler automatically arranges to repeat as much of the coding of the preceding block as necessary; then it compiles the partial block of coding, and arranges the read instructions for the tape so that this last block will overlay the repeated coding. This technique is employed to allow the maximum amount of memory space for each subsequent segment.

PART III. PRINTING THE EDITED RECORD

The printer plugboard should be a non-split, 1:1 board.

The paper loop should be a standard, 132 row loop with the following punches:

Row	*Channel*
1	7
2	1
67	7
68	1

Set the printer for normal, single space operation.

Index

Charts
- Process
  - General, 15
- Block
  - General, 18-20
  - Examples, figs. 16, 17, 29, 35
- Flow-Matic
  - General, 21-22, 26-29
  - Examples, figs. 18, 21, 22, 30, 36

Coding, Relative Machine
- General, Chapter 6, App. C
- Examples, figs. 39, 44
  - position on input tape, fig. 40

Compiler
- General, 83-90, figs. 20, 41
- Operating instructions, App. D

Data Design
- General, 23, Chap. 4
- File Design
  - General, 38-40
  - Examples, figs. 26a, 26d, 26g, 26j, 32a
- Item Design
  - General, 16-18, 41-42
  - Examples, figs. 25, 26b, 26e, 26h
- Field Design
  - General, 43-45
  - Examples, figs. 26c, 26f, 26i, 26l
- Forms, Pre-printed, App. B

Directory, 68
- Examples, 68, 70
- Position on input tape, figs. 33, 40

Edited Record Listing
- General, 88
- Example, fig. 43

Error Printouts during compilation, 108

Flow-Matic Functional Statements, Chap. 3
- General Description, 92
- Coding format, 95-99
- Guide for writing, 30-33, 93
- Examples, 19, 23, 31, 37

Format for Flow-Matic Code
- General, 93-94
- Legend, 94

Functions, description of, 92

Layout, input tape, 33, 40, 42

Operating Instructions, see Compiler

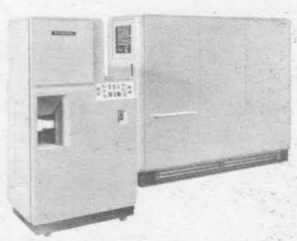
Program tape layout, fig. 42
- see Edited Record Listing

Working Storage (W-Storage), Chap. 5
- Examples, 68, 70, 71
- Position on input tape, figs. 33, 40

Univac II Systems • For data-automation which involves large volumes of input and output.

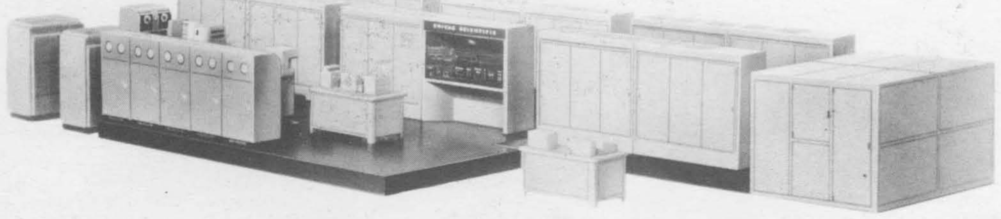
THE UNIVAC® FAMILY

Univac File-Computer • For instantaneous random access to large-scale internal storage—plus computation.



Univac 60 & 120 Computers • For speeding and simplifying the procedures of punched-card systems.

OF ELECTRONIC COMPUTERS



Univac Scientific Systems • For complex and intricate computations of engineering and research.

UNIVAC—The FIRST Name in Electronic Computing Systems

PRINTED IN U.S.A. U 1518