management standards for data processing

dick h. brandon

THE effective installation and use of data processing equipment—in ten years computer installations have grown from 1,000 to 10,000 in the United States — depends almost entirely on overall methods and performance standards to prevent costly programming waste, personnel turnover, and ineffectual performance or machine operation. Here, in this unique volume is a complete discipline of methods standards, covering the functions of systems analysis, programming and computer operation; and a complete methodology for performance standards for personnel and equipment. There is no other book now available as a guide to these aspects of computers and punched card data processing.

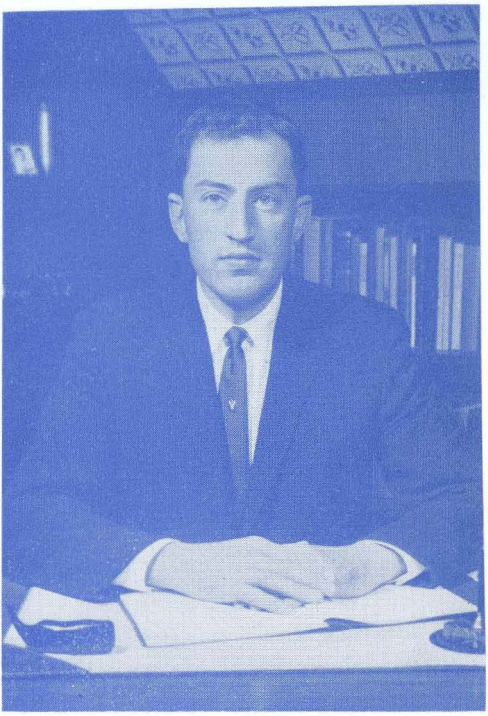
Based on the author's extensive experience in the field, the book details the specific manner in which various tasks should be undertaken and provides definite criteria and an approach to the evaluation of personnel and machine performance. A system of standard nomenclature and symbols is proposed to insure that the work of programmers and analysts is understood by all others in the installation. The normal length of time required for each task is given, as well as various measures to judge and upgrade quantity and quality of work conducted.

Throughout, appropriate illustrations are provided using forms and procedures taken from standards now in successful use. Numerous techniques for making a computer or punched card installation more effective are included, and an appendix provides a complete sample manual of standards for a typical 1401 computer installation.

Data processing managers, supervisors, analysts and computer technicians, corporate controllers and others in top management with functional responsibility will find this book an invaluable guide to implementing standards, a reference for control and improvement of systems in either business or scientific installations.

1530

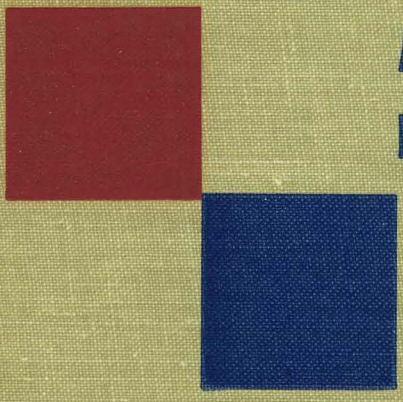
Jacket designed by George Rosenfeld



DICK H. BRANDON is President, Brandon Applied Systems, Inc., a New York consulting firm. Prior to this Mr. Brandon was Director, Data Processing Services for The Diebold Group, Inc. A graduate industrial and management engineer from Columbia University, where he received his B.S. and M.S. degrees, Mr. Brandon has taught at both Columbia and New York Universities. He was previously Director, Computer Services, Management Assistance, Inc.; Project Coordinator with IBM; and Senior Industrial Engineer and Data Processing Manager, Photocircuits Corporation. Among the professional journals to which he has contributed are the **Journal of Machine Accounting, Computers and Automation** and **Banking Magazine.** He is President, Metropolitan Computer Association and a member of the Data Processing Management Association.

1531

PRINTED IN U.S.A.



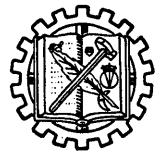
Management Standards For Data Processing

DICK H. BRANDON

MANAGEMENT STANDARDS FOR DATA PROCESSING

DICK H. BRANDON

Brandon Applied Systems, Inc.



D. VAN NOSTRAND COMPANY, INC.

PRINCETON, NEW JERSEY

TORONTO LONDON

VAN NOSTRAND REGIONAL OFFICES: *New York, Chicago, San Francisco*

D. VAN NOSTRAND COMPANY, LTD., *London*

D. VAN NOSTRAND COMPANY (Canada), LTD., *Toronto*

Published simultaneously in Canada by
D. VAN NOSTRAND COMPANY (Canada), LTD.

06675a30

PRINTED IN THE UNITED STATES OF AMERICA

TO MY FATHER

(1905-1944)

Preface

The purpose of this book is to provide a practical guide for data processing managers, supervisors, and analysts in the development of a methodology and measurement yardsticks for the operation of an ADP program. It has been written to fill a definite gap in the technology and literature, and to provide a more formal approach to a problem which to date has been largely neglected.

The growth of information technology in the last decade has been overwhelming, and has changed the scope and technical requirements of management. The installation of automatic data processing equipment has become almost a competitive necessity in this age of growing paperwork, as witnessed by the incredible number of true computers currently being installed. These installations are accompanied by mounting costs, and ever-increasing hardware capacity and complexity.

There is no question that the installation of ADP equipment is in many industries the most technical task that management has ever faced. The complexity of this technology is such that few management men have the time, inclination, or training to obtain sufficient knowledge to direct its use adequately. As a result, there has been no organization or control applied to the installation program. The absence of these, the rapid change in the technology, and the "crash program" basis on which most installations are founded, have resulted in installations of questionable efficiency and economy. The same pressures prevent the possibility of ever creating true measurement techniques, so that they are self-perpetuating. The primary objective of this book is to outline formal methods for organizing the data processing program. It provides formal guidelines for every aspect of the program and for the techniques necessary to evaluate the progress and performance on a controlled basis. The book has been designed to achieve the following:

- Provide data processing management with a definitive methodology for the installation of good standards and procedures.
- Provide the skilled data processing technician with the proper methods for organizing his own work.
- Provide top management with a guide for the continued review of progress.

Many organizations have contributed to the materials used as illustra-

tions throughout this volume and my gratitude is expressed to these organizations, specifically

The Bowery Savings Bank
General Dynamics/Astronautics
The State Street Bank and Trust Company
National Cash Register Co.
General Electric Corp.
Metropolitan Life Insurance Co.
Autographic Business Forms
Advance Data Systems, Inc.
Chrono-Log Corp.
Financial Publishing Co.
The Bulova Watch Company
Lockheed Aircraft Corporation
The U. S. Post Office
UNIVAC Division of Sperry-Rand
International Business Machines Corp.
Northwestern National Life Insurance Co.
Prudential Life Insurance Co.
Atlantic Refining Co.
Engler Instrument Co.
Electronic Associates, Inc.
Standard Instrument Corporation

In all of this work, The Diebold Group, Inc. has played a major role: it is the organization that allowed me to develop my ideas; it has been the prime source of much of the industry experience in this area; and most importantly, it consists of a group of human, incisive, and capable people. Mr. John Diebold deserves much of the credit for this, and for his major contribution to the book; Mr. Ralph Weindling, the "Exec" in many ways, has also earned my deepest appreciation and respect for his many-sided support.

My further thanks are due to a number of helpful clients, whose technical and personal support made this book possible. Mr. Ben Natchez, of Bulova Watch Company; Mr. David Thorndike, of Financial Publishing Co., Messrs. John Larsen, Peter Andre, and Arthur Hutt of the Bowery Savings Bank; Messrs. Tom Morrow and Walt Cannon of Lockheed-Georgia Company; Mr. Carl Diesen at General Dynamics/-Astronautics; and Mr. William Smith at Fireman's Fund Insurance Co., are but a few who contributed far beyond my expectations.

I have always marveled at the many ways in which authors express their gratitude to their most important contributors. In thanking my editor, associate, and critic, Mr. Arnold D. Palley, I have exhausted my resources, perhaps as he did in listening to me for many hours. The book itself is ample evidence of his contribution: he read the original manuscript in its entirety and extensive changes in style and emphasis were made in accordance with his suggestions.

And in the last analysis, as always, if words alone were used to thank my wife, Sonya, for her inspiration and assistance, I would be wholly inadequate to the task. I am lucky to have her understanding; without this there would be no preface.

DICK H. BRANDON

New York, August, 1963

Table of Contents

Chapter I

BACKGROUND OF STANDARDS DEVELOPMENT

INTRODUCTION

The use of computers to solve business or engineering problems is now commonly accepted as practical. Largely because of the rapidity of this acceptance, the data processing industry has reached economic maturity without developing proper working methods, procedures, and disciplines. This has given rise to serious operating problems, characterized by loss of management control.

The data processing profession now has the responsibility for restoring the control function to management. To do this, each computer installation, present and planned, must adopt an internal set of rules and procedures; these rules and procedures may be referred to as *management control standards.*

This book has been written to outline the method to be used in developing such rules and procedures; it includes examples and guides, as well as the reasons for the different kinds of standards that are required. This book has been written for the executive who wants to improve his data processing operations, for the data processing manager who wants to install effective standards, and for the technician who wants to understand the reasons for and benefits of a better methodology.

The primary emphasis of the book is on the establishment of standards for a computer installation, i.e. electronic data processing. The principles and techniques are equally applicable, although to a lesser extent, in punched card installations. One chapter has been devoted exclusively to the latter; the remaining chapters relate primarily to standards for stored-program computers.

DEFINITIONS

The term "standards," as used in this book, denotes a discipline which provides both guides and yardsticks:

- As guides, standards are used to establish uniform practices and common techniques
- As yardsticks, standards are used to measure the performance of the data processing function

This book describes both types of standards; the first are *methods standards:* in data processing these prescribe the complete methodology to be followed in

- Systems analysis
- Programming
- Computer operations

The second kind of standards are *performance standards*—measures that make it possible to review performance of personnel and equipment.

To illustrate the importance of *methods* standards, it is only necessary to observe a programmer trying to change a program that he did not write; the absence of uniformity of language, lack of proper documentation, and the limited organization of the program all contribute to the difficulty of understanding. The importance of *performance* standards is illustrated daily by those installations whose development budgets and schedules far exceed the original estimates. All of this can be avoided with proper management planning through the initial installation of standards. This book describes the hows and whys of standards installation.

THE GROWTH OF DATA PROCESSING

Mechanical data processing was introduced in the 1880's, when the first punched card system was developed by Dr. Hollerith. In the 1940's electronics was first used to handle large volumes of data, and in March 1951, the first commercial installation of a computer was made. At that time the United States Bureau of the Census took delivery of the first UNIVAC®, manufactured by the Remington Rand Corporation.

After that historic installation progress was rapid. Seven years later, in March 1958, over 1,250 computers had been installed and in July of 1962 this figure had increased to almost 9,500 installations, with 7,000

more on order. These figures are shown, and plotted graphically in the table below and in Fig. 1-1, respectively. An extrapolation of these curves indicates that the growth is not yet over: the installed computer population of the United States will be over 14,000 by 1965 and an estimate of such growth through 1970 shows the possibility of almost 20,000 operating installations in that year.

GROWTH OF UNITED STATES COMPUTER POPULATION *

	Total Number		Most Popular		
Date	*Installed*	*On Order*	*Unit*	*Installed*	*On Order*
March, 1958	1,277	1,601	IBM 650	750	1,200
January, 1959	2,034	1,237	IBM 650	800	1,100
January, 1960	3,612	1,364	IBM 650	1,200	150
			IBM 1401	0	750
July, 1960	4,257	4,377	IBM 650	1,280	100
			IBM 1401	0	3,000
January, 1961	4,528	6,246	IBM 650	1,090	20
			IBM 1401	75	4,000
July, 1961	5,371	7,437	IBM 1401	510	4,800
January, 1962	7,305	7,904	IBM 1401	1,750	5,200
July, 1962	9,495	7,286	IBM 1401	3,225	4,750
January, 1963	11,078	7,097	IBM 1401	4,300	4,275
		COMPARABLE FIGURES FOR EUROPE *			
April, 1962	1,359	1,301	IBM 1401	233	677

* Source: Semi-Annual Computer Census, ADP Newsletter. © 1962, 1963, ADP Company, Inc., A Division of The Diebold Group, Inc. By permission of the Publisher.

These figures have been extrapolated on the basis of current trends. One of these trends has been a recent, sharp split of computer market requirements. The two directions which the market appears to favor are toward the very small computer, renting for $2,000 to $5,000 per month, and the extremely large system, renting for $40,000 to $100,000 per month. The number of applications has been rapidly expanding, with numerous smaller companies suddenly finding practical and economic uses for computational power. Since the technology is far from exhausted, the future of the data processing industry remains extremely bright.

DATA PROCESSING PERSONNEL REQUIREMENTS

Regardless of the reason for installing a more advanced type of data processing system, the cost of installation is often disregarded and more

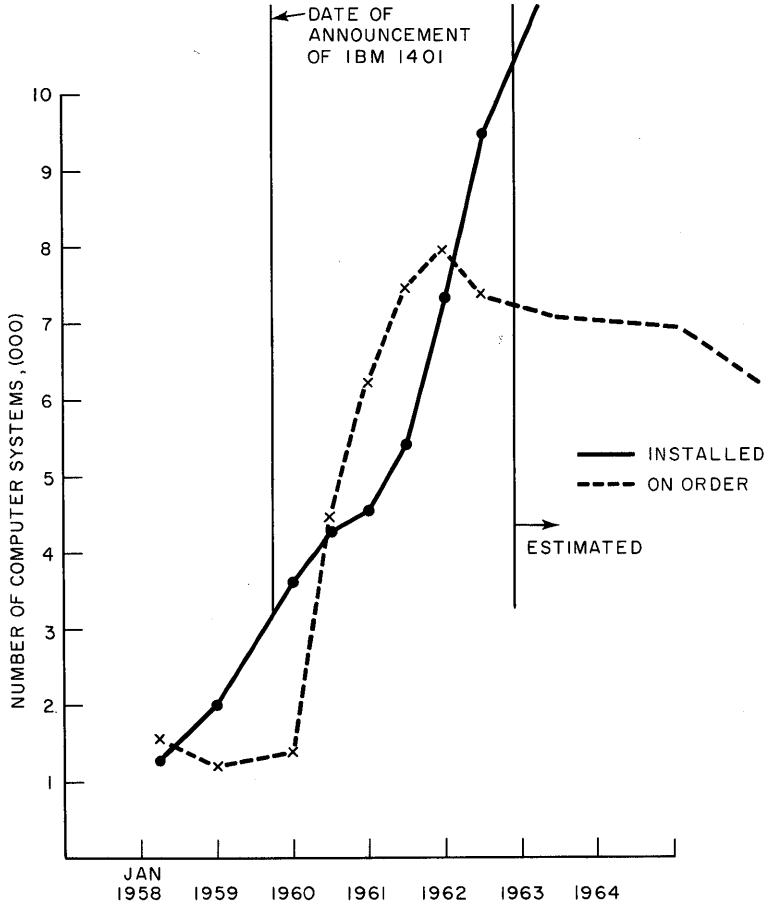


Fig. 1-1. Growth of United States Computer Population.

frequently underestimated or completely misjudged. Very few executives are fully aware of

- Their own requirements for effective management
- The difficulties of incorporating existing clerical controls into a series of computer programs
- The technical complexities of computer installation

The installation of a computer is among the most technically complex tasks that modern management must face. It requires a new set of techniques and a new set of skills, difficult to define and even more difficult to evaluate. The basic skills required in a data processing technician are:

- Aptitude: an uncanny ability to grasp all facts of a complex problem and translate this understanding into the language of the machine.
- Attitude: an outlook which insures that all conditions are provided for in the computer system.
- Interest: a consuming need to constantly refine and improve existing operations, using the most powerful tools available.

Mathematics, physics, or other advanced education are not required to become a skilled technician in commercial data processing. A college degree is not necessary if the aptitude, attitude, and interest are present. The degree is helpful in providing a certain amount of formal training; making it a prerequisite for the job is unnecessary, and markedly reduces the size of the available labor pool.

In scientific data processing, a degree in mathematics or physics is usually required. It is almost always necessary to understand the language of engineering, since most of the problems are stated in this language. The total labor pool is therefore smaller, while the demand has continued to grow.

Appendix A, page 345, Job Descriptions in Data Processing, defines the jobs that must be filled in the average installation. To illustrate the demand which is generated by each potential computer installation, Figure 1-2 indicates the average number of people required in each category for five classes of computers. These figures have been extended on the basis of the total number of systems on order and installed as of January, 1962. This extension results in an artificial requirement of 38,000 systems analysts, 65,000 programmers and 38,000 operators.* This points up the real need in the field: massive educational programs, designed to attract talented people and build a store of experience.

Because of the imbalance of supply and demand, salaries in the field have risen dramatically. The average annual salary of skilled data processing technicians has increased by almost 50% since 1958. The turnover rate in the industry has increased correspondingly, further causing severe problems in many established installations. Management has therefore been forced to the realization that *tighter control* and *management intervention* are both required in computer implementation. Management must become more aware of the problems and of the solutions and exercise direct control over every aspect of the operation.

* The extension is artificial when considering the following: There are many existing installations with more than one computer and many of the "on-order" systems will replace currently installed computers. Nonetheless, the order of magnitude is extremely realistic—especially in light of the growth since January, 1962.

It is this control that demands the establishment and enforcement of standards.

THE COST OF DATA PROCESSING DEVELOPMENT

Early installations of computers were often made on an experimental basis, with costs charged to a research and development budget. A common practice in feasibility analyses has been to ignore the development cost rather than to amortize it over a recovery period. More recently, the computer has been expected to pay its own way. Management has become aware that the true costs of installing a computer can vary from $50,000 to $25 million, depending on the complexity, scope, size, and the overall schedule.

Typical installation costs, for all elements preceding changeover are indicated below:

Cost Factor		*Low Average*	*High Average*
Site Preparation and Air Conditioning		$25,000	$250,000
Systems Analysis	4 man-years	60,000	
	15 man-years		225,000
Programming	8 man-years	80,000	
	20 man-years		200,000
Personnel Training		5,000	25,000
Machine Time for Testing		4,000	40,000
Conversion Costs		15,000	150,000
Supplies and Services		2,500	20,000
Magnetic Tapes		2,000	10,000
Contingencies		5,000	15,000
	Total	$198,500	$935,000

The cost of personnel is still the largest single cost. This cost has risen an average of 50% since 1958, so that the overall installation cost has increased by some 20-25% since that period.

This increase and the equivalent increase in operating the installation has made management more and more aware of computer costs. Accordingly, management demands a return for its money which it can only obtain under tight control. It has taken the computer out of research and development, and placed it directly under operations—so that the costs are subjected to the same management control used in other operations.

Again, the major requirement for control is standardization.

COMPUTER SYSTEM CATEGORY	AVERAGE MONTHLY RENTAL	INSTALLED AND ON ORDER JANUARY, 1962	AVERAGE PERSONNEL REQUIREMENTS FOR EACH INSTALLATION			TOTAL PERSONNEL REQUIREMENTS		
			SYSTEMS ANALYSIS AND SUPERVISION	PROGRAMMING AND SUPERVISION	OPERATION AND SUPERVISION	SYSTEMS ANALYSIS	PROGRAMMING	OPERATION
DESK	$1,800	1,865	1	1	1	1,865	1,865	1,865
SMALL	$7,500	10,645	2	4	2	21,290	42,580	21,290
MEDIUM	$15,000	1,696	4	7	5	6,784	11,872	8,480
LARGE	$35,000	904	9	10	7	8,136	9,040	6,328
EXTRA LARGE	$100,000 UP	9	15	12	10	135	108	90
TOTALS		15,119				38,210	65,465	38,053

Fig. 1-2. Estimate of Computer Personnel Requirements.

IMPLEMENTATION TASKS IN DATA PROCESSING

Installation of a computer first requires definition of the tasks that must be undertaken to achieve objectives. Unfortunately, too many computers are installed without an understanding of objectives, and many more without a definition of the tasks to be performed. The first task, either to establish a set of standards or to plan an installation, must always be to define the steps required in implementing the installation.

Figure 1-3 indicates a general time scale for the performance of the required tasks. It assumes a management-controlled development program of about 30 months; the equipment is selected in month 6 and the installation is made in month 24. The overall time will vary from installation to installation—averages have been used to illustrate the requirements. The indicated scales should not be used as a guide—an increase or decrease in the number of people used will make a sizable difference in the amount of lapsed time.

The planning and development steps are described below.

Definition of Objectives

Before the feasibility of a computer can be evaluated, management must state the objectives to be achieved by such an installation. These objectives can be indicated as dollar savings and as intangible benefits to be derived. If the objectives are limited by time, or if business can

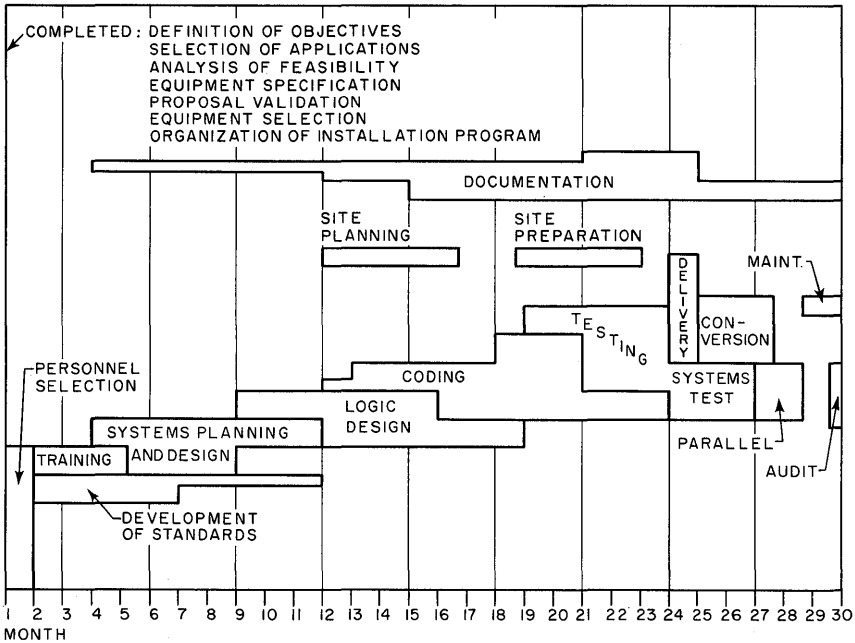


Fig. 1-3. Data Processing Tasks.

only be projected on a short range basis, management must further prepare a time schedule for the contemplated installation, so that schedule feasibility can be properly evaluated. Typical objectives include:

- A percentage reduction in the labor force, or in the amount of overtime, resulting in a minimum saving of x dollars
- Replacement of existing equipment, with an attendant cost reduction
- Reduction of space requirements
- Change from post-billing to pre-billing, thus improving cash flow
- Increase in customer service by increasing the frequency of customer statements
- Reduction in lapsed time for the preparation of quotes on new models
- Overall reduction in engineering costs
- Greater management control through exception reporting
- Edge over competition through faster determination of inventory status

Preliminary Study

After defining the objectives, a preliminary study should be made. The purpose of this study is to determine what applications should be considered, and what savings can be accomplished in each. The study will provide only general estimates; its major objective is to determine whether a detailed study is warranted. If a reasonable saving is indicated, or if other defined objectives can be met with reasonable certainty, the next step is undertaken. If the objectives are not met, the project is discontinued.

Feasibility Analysis

A complete feasibility analysis takes into account all of the costs and savings of the computer project. The study analyzes all variables and assesses the true feasibility of continuing the project. If it is feasible, the study then outlines a further plan of action and provides a detailed set of "functional specifications" from which equipment manufacturers can make proposals. The study provides a broad systems design, as well as a document that can be used ultimately to evaluate the effectiveness of the final installation.

Equipment Selection

Functional specifications have been established in the preceding step. The specifications are rarely limited to hardware because specific hardware features are relatively unimportant. The specifications will indicate the purposes which the equipment must fulfill,—for example, "the equipment must be capable of processing and updating 1 million master records per day, with an average activity rate of 7%. It should further be able to produce two sets of activity reports, with a total of 20,000 lines of print." Or, "the equipment must be able to handle the average computational requirement of a peak engineering staff of 280 engineers, each designing an average of 30 parts per year." At least three independent manufacturers should be asked to submit proposals.

It is necessary to validate the proposed equipment against the specifications. This prevents the submission of "off-the-shelf" proposals of equipment whose capability has little relationship to the requirements. It will also determine the processing time required to insure that the system is neither too large nor too small.

Selection consists of the determination and ranking of a number of factors important in achieving objectives. Each of the factors should be

given a relative weight and each manufacturer is ranked in accordance with the equipment performance. Included in these factors may be:

- Speed
- Cost
- Ability to expand or contract as requirements change
- Reliability of the manufacturer
- Systems support and other services supplied
- Value of software or subroutines supplied
- Availability of back-up equipment
- Cost of maintenance
- Compatibility with other equipment

Many other factors could be included according to individual requirements.

Selection of Personnel

Many techniques are used to select personnel for the development effort; they largely depend on the sources of personnel and policies of the organization. The group that performed the feasibility study should certainly be considered. In any case, professional selection tests and techniques should be used, just as they would be for any other professional position. The selection of data processing personnel is no different from the selection of other professionals.

Personnel Training

Several kinds of training must be organized. These include

- Company-wide orientation: to maintain employee morale
- Top management orientation: to continue the active support of top management
- Direct training: for the personnel selected as systems analysts, programmers and operators.

The manufacturer whose equipment has been selected can be of assistance in education, even though it will be limited to initial participation.

Establishment of Standards

It is vitally necessary to establish methods standardization prior to the actual start of the development program. This should take the form of a

standards manual, as discussed in detail in subsequent sections of this book. It is also necessary to establish initial performance standards to aid in determination of a meaningful schedule for the remaining tasks.

Scheduling and Coordination

In many installations, development scheduling has been done in reverse; that is, by working backwards from the delivery date of the equipment (generally from 12 to 24 months). This method unfortunately carries no guarantees since there is no correlation between the time necessary to produce the hardware (i.e. the delivery date) and the time to develop the required programs.

A better approach is to determine the actual total man-months necessary to accomplish the required tasks. If this total exceeds the time available until delivery, it is possible to postpone the delivery schedule, or to increase the staff.

Systems Planning and Design

This is the first task in the actual development program and the one least formally organized, often forgotten or done in a halfway manner. It is the development of an integrated, computer-oriented *system* of operations, designed to produce efficiently the required outputs from the available inputs. The *planning* part of the task involves the determination of requirements, analysis of the existing system, and the review of reports. The *design* function consists of developing a new system, using the computer as a central information processor. The output of the systems planning and design phase is a complete specification of the job.

Logic Design

The systems analyst has divided the system into a series of programs or "runs." Two types of logical flow charts must be prepared for each of these runs. These charts, generally referred to as *block diagrams,* completely define the internal processing logic of each of the required computer programs. The overall logic chart is called the *macro-block* diagram, depicting the total logic of the program. The detailed step-by-step elements of the program, showing all of its functions, are shown on a *micro-block* or semi-detailed diagram.

Coding

Taking the available inputs and producing the necessary outputs consists of writing computer instructions in the language of computer. This process is *coding,* consisting of the translation of a block diagram into serial instructions.

Assembly

Coding generally is not written in final machine language, primarily because the computer language normally consists of a meaningless jumble of letters and numbers. To simplify the coding task, an intermediate language is usually designed using mnemonic symbols. Thus, instead of writing the machine equivalent for each instruction, using a symbolic language one would be able to write ADD, SUB, MPY, etc. The computer is then used to translate this symbolism into its own language; that is to produce an object program. This process, the translation of the symbolic language into the machine language is called program *assembly.*

Testing

Each program is tested to validate the logical steps and to determine that all conditions have been included in the program, using fabricated test data which simulates all possible actual conditions. This function ("debugging") is one of the most important, since all programs contain errors, and since all errors must be caught. An average program contains about 1,500 separate instructions, with an average of 50 errors of different types. These errors may be clerical (in transcribing from the coding to the punched card or tape), logical, technical, or analytical.

In actual practice, of course, it is impossible to test all of the possible permutations and combinations of conditions which might arise. It is very common, therefore, to find a program that has been operating successfully for several years suddenly encounter a condition it has not met before and cause erroneous handling of a situation.

Documentation

By far the most important long-range development function is the accurate documentation of all aspects of the job. Good documentation provides complete operating instructions to meet all possible conditions and complete program instructions, so that changes can be made

independent of the person who created the program. Documentation takes place throughout all phases of the project. It is especially important during the logical design, coding, and testing phases, where it is most often neglected.

Conversion

After the programs have been fully tested in conjunction with each other, the system is ready for data conversion. This generally consists of the reformatting of existing data, the keypunching of information not currently available in machine-processable form, and the creation of tape, disc, or card files in required formats. This task may require separate conversion programs, or it may use parts of the programs written for the system. In either case, it is important to maintain exact controls over the conversion, to insure that no data is lost. Back-up information should always be retained, so that files can be reconstructed if a failure occurs, either in the conversion or in the initial operation of the live system.

Parallel Operation

The operation of dual systems is generally expensive, but insures that the new system is operating accurately. The duration of parallel operation varies from one week to as much as six months. During the initial period of parallel operation the old system is used as the prime while the new system produces secondary output. After the totals and detail items of the new system agree with the old (errors will normally be found in both) the new system becomes the prime and the old the secondary or back-up system. Subsequently, after the two agree for a reasonable length of time, the old system is discontinued.

Audit

Few currently operating installations are capable of measuring the exact dollar savings finally obtained. Nevertheless, management has a definite right to know the extent of operating economies achieved, and the degree to which the objectives of the feasibility study were met. This determination should be made as soon after installation as is practical. A further benefit of a properly designed audit program is that it may point up additional operating economies which can be achieved by changes in methods or operations.

General Organization

A definite and formal organization must be established to operate the system once it is installed. It requires the designation of control clerks, operators, librarians, shift supervisors, and the like. The entire staff requires training as part of the original program.

Physical Organization

The needs of site preparation vary according to location, type of equipment, and type of building. Some machines require raising of the floor, still others require lowering of the ceiling. Most machines, despite transistorization, need careful temperature and humidity control and require special air conditioning. A tape vault must be provided with fireproof space for extremely important files. The computer site should be isolated from other operations, to prevent unauthorized personnel from damaging the equipment or the files. The site preparation costs vary from a low of $5,000 to a high of $500,000 depending on the factors outlined above.

The Data Processing Manager

His function in the development program is of prime importance. He must set up the rules and enforce them. He must set up the schedule and coordinate the activity so as to meet it. He must guide and act as an inspiration to all members of the team and act as the decision maker in cases where a dispute occurs between an operating department and the computer development group. His superiors must continually show interest and encouragement to the program and insure that all of the operating departments cooperate to the fullest extent possible. Finally, he must install and support capable administrative and technical supervisory personnel.

Organization of the Book

The primary emphasis of the book is on commercial data processing. Nevertheless, the requirements of scientific data processing have been recognized and included. Since there is no "systems analysis" in mathematical analysis, certain sections of the book will not apply for those interested solely in the use of computers for scientific purposes. Most other sections apply equally to both areas of interest, and should be read accordingly.

The development, installation and enforcement of methods standards occupy the first part of the book, Chapters III through VII. Chapters VIII through XI concern themselves primarily with the installation and evaluation of performance standards in all areas of data processing.

For those using the book as a text, a chapter summary and a list of representative questions are included at the end of each chapter. These may be ignored by other readers.

Chapter II

THE ROLE OF STANDARDS IN MANAGEMENT CONTROL

INTRODUCTION

The term *management control* often is used to refer to the function of the corporate controller, the auditor, or the strong line executive who demands a detailed accounting for all expenses. The term when used this way is a misnomer; the intent of the user is to refer to management's control function and not to management control as such. The definition of management control used in this book is management's ability to retain complete control over the operation. Management control depends on the flow of information in a feedback cycle. The cycle consists of

- A management action
- The results of that action
- Gathering of information about the results
- Evaluation of this information by management, followed by another action where required

In applying the term management control to the operation of any department, it generally refers to the internal information system developed for the benefit of the department manager who, based on the information supplied, makes the decisions necessary to optimize the output of his department. A second responsibility of a department manager is to supply enough information to the next line of command so that the manager of a group of departments is capable of optimizing *his* output. This continues up the line, so that the manager at each level must establish his own control cycle as well as contribute to the control cycle of every level of management above him.

The data processing function, partly because of its rapid growth and partly because of its unusual technical nature, has been characterized by

a loss of management control. The manager under whom data processing falls is no longer capable of judging when an operation will be finished or should be finished. He must rely on the technicians below him, who in turn may have only a limited capability for making judgments about completion dates or costs. Because of personnel scarcity and high salaries, capable performers in data processing tend also to be temperamental—at least more so than average. As a result, they will perform their tasks in accordance with their own special conception of methods and schedules.

There is no question that the cost of this loss of control is excessive. In such instances management must take rapid action to regain control by re-establishing these control functions lost in the rush to establish electronic data processing.

The first step in achieving management control is to establish standards: standards that dictate methods of operation and standards that determine the amount of work to be produced in a given period of time. The former are methods standards—the guidelines set up to create a uniform output; the latter are performance standards—the yardsticks created to measure the performance of the staff, the department and the management. Without these standards, management control is impossible. The development, enforcement, and proper use of these standards is the largest part of the entire management control function. The remainder is merely the establishment of a schedule, a budget, and a cost accounting system. These are discussed in later chapters, in limited detail.

THE NEED FOR STANDARDS

There are a number of compelling reasons why it is necessary to have both methods standards and performance standards. Management should recognize these reasons, because management support is one of the prime prerequisites for standards establishment. Major reasons for the development of methods standards are

- A reduction of the effects of personnel turnover
- The fact that performance standards require the use of methods standards
- That economic future conversion planning can be achieved only through standardization

Reasons for the establishment of performance standards include

- The need for management control, attainable only through measurement
- The ability to develop schedules

- The ability to develop realistic costs and budgets
- The need for equitable review of personnel, and development of appropriate standards for hiring

Many other benefits will become obvious after the development program has been started.

A Reduction of the Effects of Personnel Turnover

A data processing manager who has just lost his first programmer will readily testify to the costs which he has incurred in "taking over" the inherited programs, especially if that programmer was one of the more accomplished. The language used may be highly individual, the symbology not standard, and inconsistent; the abbreviations and mnemonics absolutely not understandable. If changes to these programs are required, the problems to be faced are so severe that many programs are completely rewritten after the resignation of their authors.

In most difficult and large programs the complexity of the logic may be sufficiently difficult to confuse all but the most experienced programmer. The use of individual symbology and terminology makes interpretation difficult, and if no documentation is provided, it becomes an almost impossible task to maintain a complex program without the original programmer.

These difficulties are quite understandable. The original development schedule is almost always geared to the equipment delivery schedule. This schedule is based upon the length of time it requires to make the hardware, not on the length of time required for programming. Because no realistic performance standards are available, the required manpower is usually underestimated. The schedule then developed is based on a low manpower estimate, a meaningless delivery and completion date, and may include a lack of understanding of all of the required tasks.

As work progresses on the development program, the schedule often is not met. The schedule must therefore be adjusted, but since the completion date is firmly established in advance, the adjustment generally involves the deletion of items nearest the completion of the job. These deletions affect the last few tasks; testing and documentation are left to be completed "after delivery." Once the machine has been installed, however, the pressure from management to justify the extra cost forces the data processing department into a "crash" program to complete the testing. All other tasks, including documentation, are relegated to the category: "We'll get back to that later." The difficulty of conversion, parallel operation, and further machine justification in almost all cases prevents the occurrence of "later."

As a result of poor scheduling, and often because of a lack of detailed installation knowledge and experience, most recent computer installations have missed their target dates, and have cost well over the initial installation budgets. The original programmers who have developed the keystone programs without documentation, suddenly find themselves "indispensable," because they are the only ones capable of making the necessary changes to the programs they wrote.

To avoid this, methods standardization requires that each programmer create programs in a uniform manner, understandable to all others, with the basic minimum of documentation produced *during* the programming effort, not afterwards. This will ultimately result in much better personnel relationships; programmers are no longer indispensable—therefore they can now both be fired and promoted more easily.

Methods Standards are Required for Performance Standards

A fundamental rule of time and motion study is that the method must be completely standard before a time study can be taken. This same rule can be applied to data processing performance standards: if the method used varies from one person to the next, their output can never be equitably compared. The programmer who does not develop block diagrams will obviously complete many more pages of coding in a given unit of time than the programmer who first creates a detailed logical analysis, as documentation, prior to coding. The fact that the quality of the latter program is better, and that testing is much simplified will not affect the measurement if it is made based simply on pages of coding.

Planning for Economic Conversion

Rapid changes in technology have caused extremely rapid obsolescence in data processing equipment. It becomes impossible to guarantee that the present computer installation will be the last one, or will even last as much as five years. Many installations are already converting to a third or fourth generation of computer, having gone from an IBM 701 to a 702, 705 and 7080 or from a UNIVAC® I to a UNIVAC® II to a UNIVAC® III or 490. Farsighted management should demand that a certain amount of conversion planning is included in the planning for a current installation. This involves a recognition of the fact that programs written today may have to be converted to an incompatible machine within the next five years. The programs therefore must be written in a manner which lends itself most easily to future equipment conversion.

Some consideration of the possible conversion methods should be given in planning the development of methods standards for current programs:

1. Reanalysis.—If the new machine represents a radical departure from the current machine, reanalysis is almost always necessary. Reanalysis also represents an advantage in cases where business methods have changed, or where the specific process has not been reviewed in a long time. The reanalysis process essentially requires that the entire system be redesigned. The value of methods standards in this instance is to provide a complete description of the existing system.

2. Reprogramming from Existing Logic.—If the system is not to be redesigned the existing programs may be used, after conversion to the new machine's logic and language. To do this, the easiest approach often is to use the existing block diagrams, which will then require only coding, testing and modification of the current documentation. However, the block diagrams must be done in a standard manner and kept strictly up-to-date.

3. Reprogramming from Existing Coding.—If block diagrams are not available, or not current, it may be necessary to reprogram using the existing program listing or machine language coding. This is often very expensive, since the machine coding is directly related to the machine, and must therefore undergo major revisions. It is difficult to use this method if the coding method and program organization have not been rigidly standardized.

4. Simulation or "Machine Compatibility."—Machine-to-machine simulation, or even internal hardware compatibility devices are *inefficient* tools for an effective conversion. They are usable as "stopgap" measures, to assist in rapid initial conversion, but if the programs are not rapidly rewritten for the new machine, the new features will not be properly utilized, and the efficiency will be impaired.

5. Direct Machine Translation.—If no reanalysis is necessary, it may be feasible to accomplish direct machine translation in one of many possible forms. It is possible, for example, to develop a translation program that will translate the machine language of one machine into an approximation of the machine language of the new machine. This type of translator can never be made perfect; if 90% of the instructions can be translated directly, the remainder must still be manually reprogrammed. A similar translator translates from the symbolic language of one machine to the symbolic language of the second. A third form of translation, perhaps the most practical, uses an assembly or translation program to go from the symbolic language of the old machine to the machine language of the new machine. Although this is feasible for a higher percentage of instructions, it provides only a limited amount of documentation and "back-up."

6. The Use of A Compiler.—If the programs are written in a statement language, and a compiler is available to translate this language for the

current and the new machine, a large part of the problem of conversion has been resolved. However, high volume, high frequency programs will suffer considerably from the loss of efficiency that accompanies a statement language. (See Appendix B.) If the programs are not written in a statement language and such a language is available for the new machine, it may be feasible to translate the programs from the current listing or the block diagrams directly into the statement language; this retains the same disadvantages, while reducing the cost of reprogramming.

Each of these conversion methods requires that standardization be enforced in the writing and maintenance of the programs currently under development.

Other strong reasons for the development of uniform methods standards are:

- To enable the review of programs by a senior programmer or by a programming supervisor; it has been demonstrated to the author that the time needed for a detailed review can be reduced by more than half if the programming methods are completely standard.
- To allow segmentation of programs without encountering problems in communications between programmers. Uniform methods make it possible to divide programs among a group of programmers without concern about duplication of computer memory use.

There are similarly compelling reasons for the establishment of performance standards. These are summarized below:

The Extension of Management Control

By developing adequate personnel and equipment performance standards, it is possible for management to estimate properly the costs and time required to complete a development program, to make changes to existing systems, and to establish controls. Further, management is able to evaluate the performance of the programmer, of the program, and of the entire data processing department, against a predetermined fair standard of productivity. Management can know the capacities of equipment and of available manpower, so that appropriate resources scheduling can be done.

Scheduling of The Development Program

Without proper performance standards it is almost impossible to accurately estimate the length of time and the manpower required to

develop a planned system. This is extremely important, as has been demonstrated.

Costing of The Development Program

To make effective decisions, management must be able to determine, well in advance, the costs of the development program and the costs of making changes to an existing system. This is frequently not done, with a resultant underestimate of overall costs. So, if management is aware of the costs of systems changes, it may be possible to avoid the causes which force the changes.

Personnel Evaluation

It is necessary to make equitable adjustments to the salaries of operating and programming personnel. The data processing labor market is sufficiently competitive to force management to consider salary adjustments carefully; the loss of an experienced programmer is the loss of a large investment, and creates a very heavy cost for program "takeover." It is good business to compensate each staff member in accordance with his contribution; a good programmer should therefore be compensated more. The difficulty has been that without performance standards it is hard to recognize the exact relationships between the outputs of different programmers.

Performance standards are also necessary in the hiring of experienced personnel and in the training of inexperienced personnel. In the former case, it is necessary to evaluate the extent of claimed experience and the amount of productivity which reasonably can be expected for this experience level. In the latter case, promotion to a new grade or a change of status from trainee to junior requires a minimum performance standard which must be achieved.

It is perfectly logical that hardware is built with a rigid set of standards, covering everything from blueprint symbology to the color coding scheme for resistors. It is equally logical that the same standardization be applied to "software" production, both by the manufacturer and the user. The fact that standardization is enforced is more important than the particular standards used; if the importance of standards is recognized, their maintenance will be made the responsibility of the proper authority: the managers who most benefit from the installation.

BENEFITS OF STANDARDIZATION

The benefits which accrue to the user of good methods and performance standards are almost immediately obvious, so that a direct measurement can be made of the return which will be obtained from the investment in standards development. Among the measurable benefits are:

- A reduction of overall costs. The cost of personnel turnover, and the cost of training new personnel are markedly reduced. The more effective machine utilization which always results from enforcement techniques alone more than pays for the investment.
- Increased control over personnel, machines and facility use.
- Improved quality of output, by incorporating quality measures right along with measures of performance.
- Reduced dependence on individuals, by incorporation of uniform methods and practices. Absenteeism no longer prevents changes from being made or progress from continuing.
- The improvement of overall management techniques. By including evaluation techniques in the standards program, management obtains the ability to schedule, control and manage the program.
- Reduction of future costs. Program change is simplified, and the cost of a potential conversion is reduced considerably.
- Appropriate resources planning. Personnel requirements can be met scientifically, and not emotionally, through training, upgrading, promotion *and* the proper requirements analysis.
- Corporate long-range planning. Planning for the future can be assisted by the knowledge of future data processing capabilities and costs.

All of these factors point to the need for the development of comprehensive national, or even international standards for the effective development, utilization and operation of data processing equipment. These standards although necessary will take a long time to reach fruition. In the meantime, each installation must of its own accord develop the standards necessary for its own survival.

SOURCES OF STANDARDS

It is difficult to develop standards for data processing without recognition of the many differences which have grown up among machines, manufacturers, and industries. Within machine types, for example, dis-

tinctions are made between "scientific, or engineering" and "business, or commercial" data processing equipment, analog and digital, large and small, tape and card, alphameric, decimal, and binary, fixed word and variable, etc. For each of these different equipment systems, it may be possible to develop standards which cannot be applied to the other categories. Similarly, it may be difficult to cross manufacturer lines, since each manufacturer has his own ideas of the standard which is best. For example, IBM suggests that the symbol on the left (Fig. 2-1) be used to indicate a "decision" function; the Univac Division of Sperry Rand suggests the symbol on the right for the same function. Major differences exist:

- Across machine lines
- Across manufacturer lines
- Across industry lines
- Across user lines
- Across departmental lines, for a given user.

Other influences affect the standards program; a specific template may be established as official symbology for block diagramming for an installation. Because programming templates have undergone changes over the years, the use of an aged template is a status symbol which identifies the programmer as one of the "old-time" group.

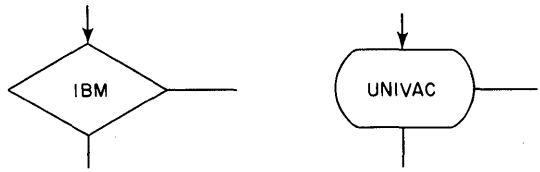


Fig. 2-1.

Similarly, the hiring of experienced programmers from outside the organization introduces a new set of experiences, practices, and rules into the organization. There presently exists an installation where among twenty programmers there are three different methods of writing the alphabetic character O.

Many organizations are currently concerned with the establishment of standards in the data processing industry. Governmental bodies, for example, recognize the importance of standardization; their primary concern is with standardization of hardware across manufacturer lines, since the Government must deal with many vendors. The international standards groups established by the various data processing organizations are also extremely concerned; their concern centers on the differences

in languages and approaches used in various countries. Manufacturers are very concerned with standardization; their interest is limited to their own hardware and sometimes the continued use of their own products.

The groups listed below are developing standards in the data processing field:

The United States Government

The Government spends in excess of 600 million dollars per year for data processing. The Government must purchase or lease equipment from all acceptable vendors, so that their main interest is in the development of compatible hardware and software. The COBOL effort, sparked by a Federal agency, and the policy which requires COBOL to be implemented by all vendors are outgrowths of this concern. The Government is just beginning to standardize its methods of development and operation; the development of performance standards for equipment and personnel will have to follow.

The American Standards Association and the International Standards Organization

These groups are currently quite concerned with the development of standards to permit hardware compatibility among all users in all countries. Some work is also being done to develop a standard glossary and to develop a standard character set to be used by all manufacturers, but implementation is slow, cooperation is limited, and the work is at the present time wholly inadequate. The basic requirements of systems analysis, programming and machine operation are not presently served by these efforts. The difficulties which have been encountered so far in the standardization of the simplest items makes it hard to foresee the time when all methods of operation will be subject to *one* international standard.

Trade Organizations

The Association for Computing Machinery, the *Systems and Procedures Association,* the *Data Processing Management Association,* and many other professional societies are attempting to standardize some areas of interest. Notable among these piecemeal efforts are the ACM standard template (which has not yet been universally adopted), and the DPMA Certificate in Data Processing, establishing a "common" standard for hiring.

Manufacturers

Most manufacturers have been long aware that the best interests of the data processing industry would be served by a cooperative standards effort. The manufacturers have generally felt that it was up to the user to develop standard practices. Some efforts are beginning to be made, but this source of standards will remain limited in its scope.

User Groups

Many users have formed associations, grouped by machine type. These groups generally exchange programs, information, and a general interest in the problems and successes of each installation. Some standards have been established, primarily to enable the exchange of programs. In these cases, the standards have indicated the documentation which must be supplied, and the forms which are to be used. Examples of user's groups which have developed such standards are SHARE (for the IBM 704/709/7090/7094/7040/7044), GUIDE (for the 705/7080/7070) and the UNIVAC® users associations.

Industry Associations

Within certain industries trade associations have themselves recognized the need for standardization. Thus, the *Aerospace Industry Association* monitors the APT III program for numerical tool control, and the *American Bankers Association* established the E13B type font for the magnetic encoding of all checks to be issued by banks. These standards are useful and necessary, but they do not usually go deep into the needs of the development program.

Management Consultants

Several major management consulting firms have developed complete installation standards in the interest of their clients. Since these standards are not freely available to the entire industry, this source of standards, although representing extremely competent and experienced personnel, will not solve the industry's problems.

The final group currently developing standards are the users themselves. Most installations have recognized the need and are implementing some type of standards program. Throughout this text, examples of user's standards will be indicated based on experience garnered in many installations. The organizations which have contributed to make this possible are listed in the Preface.

THE MANUAL OF STANDARDS

A compilation of methods standards and the rules necessary to establish meaningful performance standards is required in each installation. This compilation should be made and enforced by the establishment of a Standards Manual (or a Manual of Standard Operating Practices) relating specifically to the Data Processing Department.

The manual of standards should consist of at least five major sections, each devoted to an area of special interest. These are:

- Systems analysis standards
- Programming standards
- Operating standards
- Performance standards
- Personnel standards

If a punched card installation is required for peripheral assistance to the computer, a separate section should be devoted to those standards applicable to punched card operation.

The main functions of the manual are:

- To serve as a compilation of all rules and regulations governing the operation of the department
- To serve as a manual of policy, for reference by all employees in determining the methods to be used
- To serve as a training manual for all new employees
- To enable management to review the performance of the staff in accordance with the established policy
- To settle disputes about procedure

The manual of standards must therefore prescribe all of the procedures to be followed, the methods to be used, and the information to be produced. It should enforce rules and regulations and at the same time insure complete uniformity of output. Appendix D illustrates a manual of standards developed for a small computer installation.

THE EFFECT OF LANGUAGE STANDARDIZATION

The industry, at the present time, is embroiled in a major controversy over the use of "common" statement languages. The major languages which are being considered as common standard languages include

- COBOL—the COmmon Business Oriented Language
- ALGOL—the ALGOrithmic Language
- FORTRAN—the FORmula TRANslation language
- JOVIAL—a separate version of the International Algorithmic Language
- NELIAC—another dialect of the same language

Language standardization is indeed of great benefit to the user. If a specific compiler is implemented for all types of machines, all programs written in this language could in theory operate on any machine. This prevents a major problem in conversion planning and enhances the chances of lesser known computer manufacturers. The statement languages above have other advantages; a major claim made for the statement languages is that they are far simpler to learn and far simpler to understand than the complex symbolic or lower level languages. Proponents of the statement language claim that the language is in itself sufficient documentation of the program; no further block diagrams or other explanation is required.

Some of these claims are overstated, and perhaps even damaging to the common language effort. Management does not always recognize the problems that data processing implementation presents. If the advertising claims made by manufacturers (ascribing magical powers to the higher level languages) are misunderstood by management the data processing effort may well be seriously impaired. The higher level languages still require block diagramming; they also require considerable added documentation, if they are to be properly understood by programmers and operators alike. Learning of the higher level language is indeed simpler, but a good programmer is still required to effectively use such a language, so that programming as a skill is far from eliminated. In addition, the most common disadvantages of these higher level languages is that they seriously increase program compiling time and considerably reduce the efficiency of the object program.

Appendix B illustrates the approximate relationships between the costs and the efficiency of higher level statement languages. The Appendix also indicates that these languages will gain enormously in their acceptance in the next few years, but that symbolic languages will still be required to process the smaller group of "bread and butter" programs.

It is not true, however, that the increased acceptance of these statement languages will reduce the need for the complex methods standards outlined in this text. The tasks of programming and systems analysis will change very little; it will still be necessary to develop detailed logical specifications, detailed block diagrams and other documentation. The only change indicated by the use of a higher level language is in the

coding method: instead of coding in a rigidly restricted, single line symbolic notation the coding format is free-form, statement-like commands. This additional freedom is not necessarily constructive; the increased freedom which has been given to programmers has been one of the principal causes of the current problems of communications among programmers. It is therefore necessary that the same rigid standards applied to symbolic languages be included in the higher level languages to prevent a recurrence of the problems currently plaguing the industry.

STANDARDS AND CREATIVITY

Most accomplished programmers are certain that programming is an art, not a science. This thinking is applied with equal force to systems analysts, and sometimes even to operators, who may be referred to as "playing the console like a piano." It is unfortunate that the artistic, in business, is usually regarded as remote from the profit motive; thus the qualified data processing analyst must justify his "art" as a science.

Many experienced personnel argue against the establishment of standards, much as they might argue against the benefits of documentation, or the value of "closed shop" program testing. A common myth among "creative" people is their feeling that creativity is inhibited by rules and regulations. This is entirely false. Good standards and good practices never inhibit creativity; creativity is greatly enhanced by good operating procedures. A diabolically clever programmer is far more inhibited by a lack of appreciation of the value of the understanding that he must leave behind him. Channeled and controlled by effective rules, the programmer will increase his productivity and his own understanding.

A good programmer retains his awareness of the *overall* objective: to produce a quality program in a given amount of time, minimizing the corollary costs of testing, documentation, and parallel operation, and optimizing the ability of others to understand the functions and workings of his program. This individual will contribute most to the development program, and will rapidly recognize the need for and value of a complete standards program.

Good standards enforce themselves. Once the programmer recognizes that his own performance is improved by methods standardization he is its foremost proponent. When he suddenly recognizes that he is capable of understanding a program written by someone else, he is convinced forever. The author has seen many cases where experienced programmers rebelled at the entire concept, but once forced, they recognized the benefits derived, asssisted in further development, and helped to enforce standards with the remainder of the staff.

ORGANIZATION STRUCTURE OF DATA PROCESSING

To avoid many of the problems of ineffective data processing, *top* management must consider the following at the time it initiates a conversion to electronic data processing:

- It must obtain a realistic education in at least the basic principles and planning steps of data processing.
- It must select the data processing manager on the basis of his ability to act as a manager, not on the basis of tenure. He must be capable as a technician, an administrator, and as a salesman of good practice.
- It must establish an internal data processing organization structure that lends itself to effective administration.
- It must enthusiastically support the development of methods and performance standards.
- It must require a standard method of progress reporting, enabling a rapid recognition of schedule slippage or poor performance.
- It must, in advance, provide for the establishment of alternatives to be available in the event of schedule difficulties.

An outline of a basic organization structure is given as Figure 2-2. The data processing department is divided into three major sections: systems analysis, programming, and operation, each under a supervisor or manager who reports to the data processing manager. There are also two major control functions in the department, each headed by an assistant manager: the first function might be regarded as "development" control, the second as "operating" control. Neither of the two control functions report to the operating managers whom they serve; to preserve independence they should both report to the data processing manager.

Regardless of the size of the organization, the manager of data processing should either report to or be an officer of the Corporation. His reporting function should be outside of the control of the major user; that is, an engineering data processing center should not report to the chief engineer, and a financial data processing center should probably not report to the Vice President, Finance. The most logical place for data processing is as a separate arm of management services, with sufficient independence so that all users will receive equal service, yet sufficiently high on the organization chart to command the support of top management to be able to develop its ultimate potential.

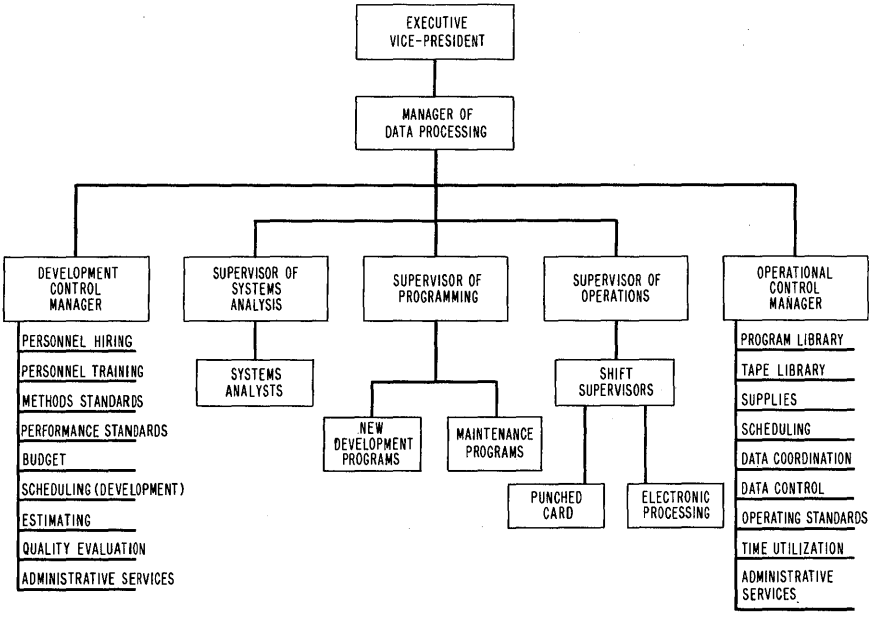


Fig. 2-2. Organization for Data Processing.

CHAPTER SUMMARY

Chapter II has developed the relationship in data processing between the concept of management control and the concept of standards. The major needs for standardization lie in the areas of methods and procedures, which today are often left to the discretion of the programmers, analysts and operators. Management's need for control is the most significant reason for the establishment of a good standards program. There are other reasons; included in these are the necessity for conversion planning, the need to reduce dependence on key members of the staff, and to make promotion possible, and the need for effective scheduling and cost estimating.

It is recommended (and the entire text assumes) that a Manual of Data Processing Standards be established. Included in the manual will be all required good practices and all formulas and relationships for performance measurement. Management control *can* be applied to creative people, and will assist in channeling the creativity into the most productive channels.

A separate and distinct organization structure is recommended for data processing. It includes a powerful and competent data processing manager, and a five-part organization reporting to him. The three line functions are systems analysis, programming and operations; the staff

functions are those control functions required to operate effectively and economically.

Questions for Review

1. What is the most important problem in data processing today? Discuss this in a two page essay.
2. What are the major reasons for installing performance standards? What objections would you expect from the staff?
3. What are the major reasons for installing methods standards? Do you expect similar objections from the staff? How would you counter such objections?
4. List the areas which you feel could be easily standardized in a data processing department. Which of these areas do you feel should be developed by the manufacturer? by the Government? by the users groups? by others?
5. What are the basic differences between performance and methods standards?
6. Do you feel that management control could be established without standards?
7. Do you think you could derive performance measures for programming and systems analysis without first standardizing the method? Why or why not?
8. Discuss the effect you feel will be felt by the development of complete language standardization; indicate the differences which would be necessary in a standards program developed for an installation working in symbolic, and for another installation working in a statement level language.
9. Indicate the most valuable standards which could be derived by the equipment manufacturer.

Chapter III

METHODS STANDARDS: SYSTEMS ANALYSIS

INTRODUCTION

Systems analysis represents a major link in the chain of translation from the problem to its machine solution. Because methods standardization is a vital and necessary task in this translation, it follows that there is a great deal of potential for standardization in the systems analysis function. Perhaps more than any other function, systems analysis relies on creativity, rather than rote analysis, to develop effective computer systems. But, this creativity must be channeled and documented effectively, if *lasting* value is to be obtained.

THE SCOPE OF SYSTEMS ANALYSIS

Man-machine communication is the term that can be used to describe the process required to translate a problem into a form suitable for machine–assisted solution. This communication process may be done by one person as in the case of the engineer who writes his own FORTRAN program, creates the data, and develops the solution. More commonly the process consists of several functions, ranging from problem analysis to machine operation.

Figure 3-1 shows this process in the three most frequently found arrangements. Because of cost, the organizations are often related to computer size; the largest computer has the most functional organization, with seven distinct functions; the smaller system has only two functions and may have only one. In each chart, however, the basic elements that must be performed are reflected in the "average" case. The functions are outlined below.

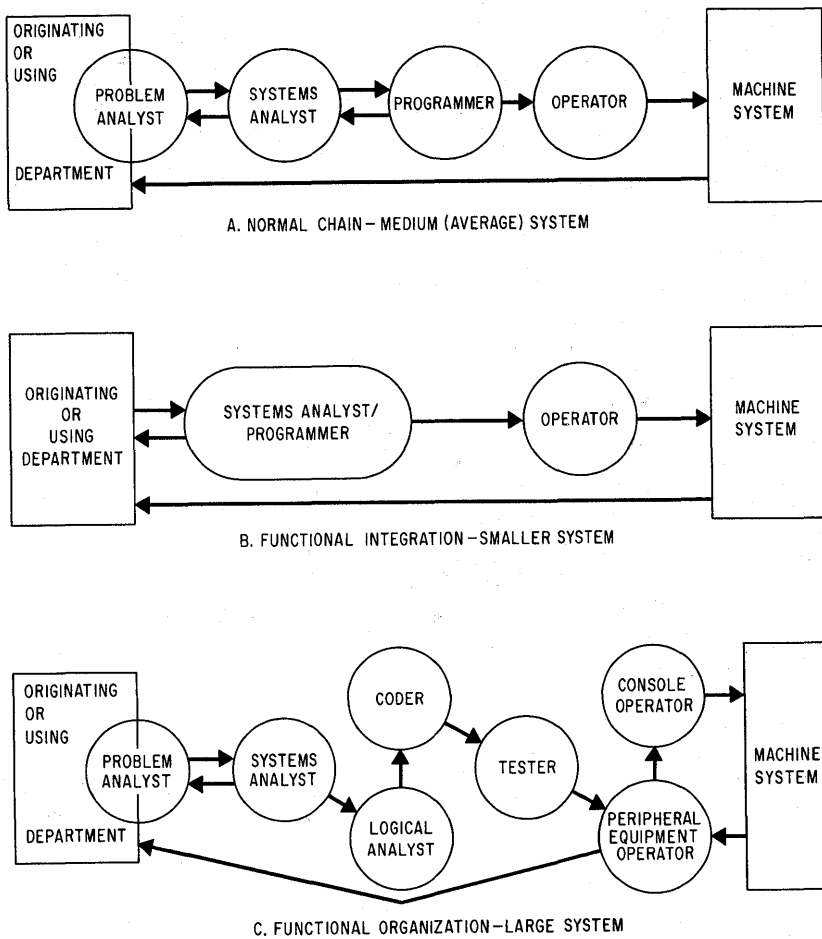


Fig. 3-1. Typical Functional Flow of Design.

- Problem analysis—This function consists of defining the problem and of determining exactly what is required in the solution. It is generally performed by an expert in the application, who may be a member of the using department.
- Systems analysis—After the problem and its requirements for solution have been stated in clear terms, the systems analyst defines the broad outlines of the machine solution. He must know the overall capacities of the equipment, and he must be familiar with the application. His specification of the problem serves as the link between the problem analyst and the next function, programming.

- Programming—The defined machine solution is translated into the language of the machine by a programmer, who creates the instructions and validates their accuracy. The programmer must be a machine expert but he need not have a detailed understanding of the application once the analysis is complete.
- Operation—The operation of the machine creates the output information from the designed inputs. The operator has no knowledge of the application or the program; he only need be capable of operating the console and of reading the documentation provided.

Using a payroll problem as an illustration, the following functions would be performed by each of the four areas of responsibility.

Problem Analyst: A personnel expert assigned to the payroll department; possibly an accountant currently responsible for the maintenance of controls and audits in the payroll area.

Functions:

To state the objectives of a mechanized payroll system

To design the required outputs; i.e., indicate what fields should appear on each document; with what frequency each document should be produced

To indicate the inputs; design worksheets, or other forms suitable for machine processing or for key translation; indicate which information is variable, which fixed

To trace exception conditions, and define their processing methods

To determine formulas presently used, and provide them in an understandable manner

To indicate the types of controls desired and mandatory in the processing

To act as general liaison between the data processing section and the payroll area

To approve the job specification, if complete and satisfactory

To analyze document necessity

To supply test data for testing the overall system

Systems Analyst: A data processing expert, familiar with the equipment and its capacities.

Functions:

To develop the job specification manual

To develop input layouts for machine processing

To develop output layouts in machine formats

To develop layouts of records to be retained
To create the overall flow of information through the system, in order to retain and create all of the desired information
To develop individual specifications on each of the programs making up the payroll run; including in each specification:
- a timing analysis
- a brief description of the program
- an input-output flowchart
- a statement of the functions of the program
- a description of the features of the program
- an indication of the formulas to be used
- a statement of the controls and audit trails to be maintained in the program

To assist the problem analyst in further improving the system, wherever possible
To prepare overall test data to validate the programs
To assist the programmer in all aspects

Programmer: A machine expert, familiar with the machine's language, and with logical analysis.

Functions:
To develop the overall program logic of each run
To develop the detailed logic, without forgetting any exception conditions which might occur
To develop a program of instructions, coded in a symbolic language to enable the machine to execute the functions defined
To develop a set of data, sufficiently comprehensive to test a maximum number of conditions included in the program
To actually operate the program under testing conditions on simulated data, in order to eliminate all possible errors
To develop sufficiently comprehensive documentation so that both an operator and another programmer can run and maintain the program in the future without further assistance
To assist in training of the operator
To assist in conversion and parallel operation

Operator: An expert in the operation of the equipment generally unfamiliar with the details of programming.

Functions:
To set up the machine for processing of the payroll programs

To initiate the machine's functions
To observe the normal processing to avoid undetected equipment malfunction
To act under the general direction of the program, its manuals and the supervisor to correct machine, data, program or operator errors which occur
To "take down" the system at completion and dispose of all materials, returning tapes to the library in a protected manner, maintaining output documents and cards, and refiling the program deck or tape
To maintain good housekeeping in the machine room

The same general functions will be found in any other data processing problem. Some slight variation may be found in the manner of processing an engineering or scientific problem, but the basic concepts remain the same. In an engineering problem, the problem analyst or engineer will frequently also act in the capacity of systems analyst. This is directly due to the complexity of these problems; they require a complete knowledge of the application and its most effective mathematical solution. It is further possible for the engineer or scientist, in an "open shop," to do his own programming. The engineer need not be an expert in the machine; the manufacturers of most systems provide a type of "software" for engineering and scientific problems one level above the symbolic level, such as FORTRAN.

The systems analyst is basically responsible for defining the machine-oriented variables in any problem. He must provide the translative link between the problem definition and the ultimate program of instructions for the machine. The "job specification manual," signed and approved by the using department, serves as this link.

STANDARDS IN SYSTEMS ANALYSIS

The functions to be standardized in systems analysis include:

- Definition of terms: the development of a glossary of terms and abbreviations necessary in the installation
- Layout: the method with which card designs, report formats, tape records and the like are displayed in graphic form
- Procedure and document analysis: the review of output and input documents
- Problem definition: definition of the problem and its requirements
- Control coding: the assignment of meaningful numbers to programs, systems, reports, files, and the like

- Flowcharting: the pictorial display of a process or system
- The job specification manual, the final documentary output of systems analysis

These are discussed below.

Definition of Terms

The most basic standard for any installation is the proper definition of terms. It is quite common to find descriptive terms with more than one meaning used in system documentation. The term "label" to an operator means the external pressure-sensitive label used on tape reels; the same term used by a coder means the label or tag used to identify an instruction or routine; when used by a systems analyst or programmer it may well mean the magnetic tape label that is always the first record on tape. It is therefore quite important to establish a fairly rigid dictionary of terms for use throughout the installation. This dictionary should consist of three parts:

Part I. Glossary of Data Processing Terms.—The glossary of data processing terms should consist of a definition of common data processing terms, such as adder, accumulator, address, binary, buffer, debugging, etc. A very brief sample glossary is reproduced as Figure 3-2, to indicate the simplicity of the definitions. More detailed glossaries may be obtained from the Association for Computing Machinery or from any computer manufacturer. Glossaries have also been published in a number of recent texts and can be reproduced in a standards manual with the permission of the publisher.

Access Time—The time needed to read or write data from or to storage; that is, the time between the request for information and its delivery.

Accumulator—A computer register where numbers are totalled. It is normally contained in the arithmetic unit.

Adder—A device within the arithmetic unit capable of forming the sum of two quantities. This may be a one-bit, one-word, or one-character device.

Address—The label of a storage location, register location, or specific input/output device, which enables reference by the program.

Analog Computer—A computer which uses electric or electronic pulses to simulate a physical system, rather than using digital codes.

Automation—The application of self-activating machines to the control of production, processing, or the manipulation of business data.

Binary—A number system using as a base the number 2. Only two states are possible in this system: 0 and 1. This is the number system used by all digital computers.

Bit—An abbreviation of binary digit; either 0 or 1 in the binary system.

Buffer—A device used to help equalize differing speeds of two computer

components. A buffer loads at the rate of the lower speed unit while the higher speed unit is occupied with some other process. The buffer unloads at the speed of the faster unit.

Code—Symbolic representation of data or computer instructions, necessary to translate human language into computer language.

Command—An instruction to the computer to perform one of the specific operations it is capable of executing.

Compiler—A special routine or computer program used to produce a set of specific computer commands from generalized statements.

Computer—A device capable of performing sequences of internally stored instructions, including arithmetic or computational functions. Commonly a data processing system controlled by this device.

Cybernetics—The study of control and communications in man and machines. Cybernetics research is directed towards electronic duplication of human brain mechanisms.

Debugging—See *Testing*.

Digital Computer—A computer which uses discrete symbols and integral values to process information.

File Maintenance—Processing of a master file of records, on a regular schedule, to make changes, adjustments, insertions, and deletions.

Flow Chart—A diagram of the sequence of processing steps, using a set of conventional symbols.

Hardware—The mechanical and electronic parts which make up a data processing system.

Information Technology—The science of using a computer to process information.

Input—Information transferred from an outside source to the central computer processor.

Instruction—A set of characters or bits which defines a computer command.

Integrated Data Processing—The combination of communications equipment with data processors to link remote locations or an information system.

Label—(1) A magnetic tape label written as the first, identifying, record on a tape file; (2) An *external* label refers to the pressure sensitive label required to mark a reel of tape; (3) A *tag* is used to define the label or name of a program step.

Magnetic Core—A small ferrite ring capable of storing the *on* or *off* condition of a bit; the basic storage element of most solid state computers.

Magnetic Drum — A rotating metal drum on which information can be stored as small magnetized spots representing bits.

Magnetic Tape—A continuous strip of plastic or steel, coated with a magnetizing substance upon which data bits can be written.

Memory—See Storage.

Millisecond—one thousandth of a second (0.001 seconds).

Microsecond—One millionth of a second (0.000001 seconds).

Nanosecond—One billionth of a second (0.000000001 seconds); sometimes called milli-microsecond.

Output—Information transferred from the central computer processor to an outside device.

Program—The complete sequence of instructions necessary for the computer to complete a process or solve a problem.

Punched Card—A rectangular cardboard card upon which data is recorded using a punched hole code.

Random Access—The ability to bring data from any location in storage in the same access time.

Routine—A set of computer instructions which carry out some well defined function; usually part of a program but sometimes used to designate an entire program.

Software—A set of programs and routines supplied by the manufacturer

of a computer, used to handle processes common to all users.

Standard—In data processing (1) a guide, as a methods standard, for programming or operation; (2) a yardstick as in performance standard.

Storage—A device capable of retaining data and supplying the data on command. Internal (or working) storage contains the computer program, the data being worked on, and constants. External storage is removable from the computer and holds data in a form acceptable to the computer. Auxiliary (or file) storage is attached to the computer and supplies frequently needed information to the internal storage.

System—(1) A collection of "hardware," integrated to perform functions, as in a data processing system. (2) A collection of programs and procedures, to perform a specific application, as in accounting system.

Testing—The process of determining the correctness of a computer routine, program or systems application.

Word—A set of continuous bits or characters, read, written, and transported by the computer as a unit.

Fig. 3-2. Glossary of Data Processing Terms. (*Courtesy*, The Diebold Group, Inc.)

Part II. Glossary of Industry Terms.—The second part of the dictionary should be a short list of terms peculiar to the industry or company in which this material is to be used. Programmers are often hired with little or no experience in the specific industry or company. To reduce communications problems a fairly detailed glossary of terms can be extremely useful.

Figure 3-3 illustrates part of a glossary of terms designed for the brokerage industry.

Definition and abbreviation *	
Blotter * BLOT	A daily record of activity, broken down by type, i.e., New York Stock Exchange odd lots would be one type of activity.
Bond * BOND	A debenture or debt obligation of a corporation or municipality. Generally referred to by its designating kind, as * MUN-a municipal (bond) issued by a town or county * CORP-a corporate (bond) issued by a corporation * CONV-a convertible (bond) capable of conversion into stock * FORN-a foreign (bond) issued by a foreign power or municipality * GOVT-a government (bond) issued by the United States Treasury
Bookkeeping * BOOKS	The daily balancing of cash, including the interest income earned on margin accounts.

Coupon * CPN	A section of a bond maturing at regular (usually six months) intervals, used to pay the interest charge. Also synonymous with interest distribution.
Dividend * DVDND	A distribution of part of the income of a corporation to its stock holders (in cash—a cash dividend; in stock —a stock dividend).
Fail * FAIL	The failing of a delivery or a receipt of stock, due to be received or delivered within four business days after its trading.
Margin * MARG	The purchase of stock using part cash and part credit. The credit amount used is charged with interest, and the stock is held in safekeeping as collateral against the loan.
Odd-lot * ODLT	The purchase of a lot of stock in less than the amount normally traded. (The normal trading amount for most stocks is 100 shares; some stocks are traded in lots of 10, 25 or 50 shares.)
P & S * PS	Purchase and sales of stocks or bonds—the lifeblood of the brokerage business.
Round Lot * RNDLT	A lot of stock in integral multiples of its normal trading of 100 shares, except as noted under odd lot.
Stock * STK	The equity or part ownership of a corporation.
Stock record * SR	In a brokerage firm, the total list of all securities held by its customers, in order by security.
S/R Take-off * SRTOF	The activity and changes to the Stock Record (S/R) made on a daily basis. Such activity is based on the sales, purchases, receipts and deliveries of stock from and to the customer.

Fig. 3-3. Excerpt from a Glossary of Stock Brokerage Terms.

The principle of having a glossary for industry terms can be further extended to each of the applications to be installed. The first part of the assignment of a problem analyst should be to construct a brief glossary of all terms in the application which are used, with a specific meaning other than that in the dictionary. This is a powerful technique which has the benefit of forcing the analyst to define his terms, create the appropriate documentation, and specify exact meanings for all the processes and data being handled.

In addition to the definition which appears for each term in the glossary, there should be a standard *abbreviation,* or notation, for any term which is subject to being shortened for use in block diagrams, coding, or other documentation. This abbreviation should apply to almost all defined terms for the industry and the application, and its usage should be enforced.

Part III. Specification of Terms.—General rules which apply to the

computer, its objectives and its use are referred to as a "specification of terms." These rules should also appear in the first section of a standards manual, because they cross application lines, and apply to *all* of the functions involved. (Rules and regulations which apply only to specific functions are developed in the appropriate chapter of the manual.)

The following rules might appear in such a specification:

1. All tapes used by the installation shall carry magnetic tape labels having the following format:

Label Identifier
File Number
File Description
Creation Date
Retention Cycle
Reel Number
Tape Reel Inventory Number

2. All tape files shall have a sentinel which follows the label and precedes the first data record.

3. All files shall carry a trailer label which follows the last data record and carries an indication of whether this reel is the last reel of the file. The trailer label will also carry a "hash" total of the first five characters of each tape data record (not including the label), and a record count.

4. All card records shall carry an identifying punch in column 80, as follows:

0 Normal data input
1 Standard date card
2 End of data sentinel
3 Program card
4 Special parameter card

5. All programs shall be loaded from a Master Program Tape, which shall be updated on a weekly basis.

6. No more than one file of data shall be placed on tape; there shall be no multi-file reels used.

7. The standard input-output system shall be used in all programs, without exception.

Layout Standards

One of the responsibilities of the systems analyst is to develop complete layout records for all inputs and outputs of each program. This will include layouts for

- Card input and output
- Printer output

- Tape input and output
- Memory layouts in cases where the use of special tables is specified.

For each of these categories standards should be developed which provide a standardized format for the layout, and an indication of the detail required.

Card Layout Standards.—Most manufacturers provide card layout forms to prepare the necessary electrotypes for the printing of the card. In addition to this, for documentation purposes, it is useful to indicate field names, field sizes and field characteristics. Suggested card layout standards are given below:

1. A separate card layout shall be prepared for each card format used by the program.
2. The card layout shall be made on standard form No. *XXX,* and shall include

The application
The program name and number
The card volume anticipated
The layout of the card, and for each field
the field source
the field size
the data type
the columns
the field name or description, and its abbreviation.

3. If more than one card format is used in the program, a multiple card layout shall be made in addition to the detailed layouts for each card.
4. The multiple card layout shall be made on form No. *XXX,* and shall reference each detail card layout with its card number.
5. Vertical lines shall be drawn to show the field definitions and horizontal lines to show interpreter fields and field names.
6. For purposes of input validation, the field type shall be indicated to be

Alphabetic
Unsigned numeric
Signed numeric
Leading zeroes/blanks

7. For purposes of consistency checking, field limits should be shown on the layout. For numeric fields this should be a range, or a list of allowable codes. If there is no consistency checking possible, it should be so indicated.

A sample card layout form is shown as Figure 3-4. It was printed from the manufacturer-supplied card layout form, which was reduced and superimposed over the form. A similar form for dual purpose cards was made up by superimposing a blank card layout on the same background.

A multiple card layout form is shown as Figure 3-5.

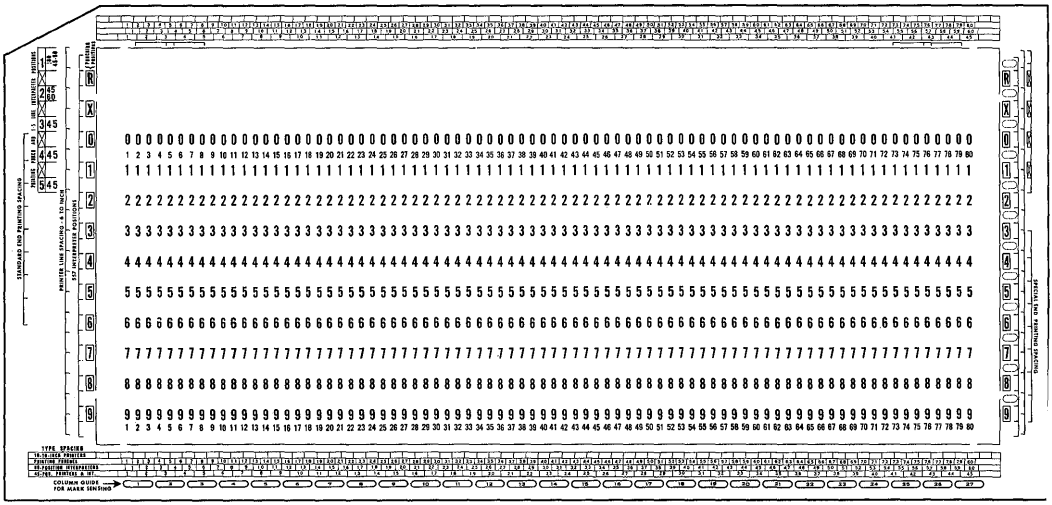
BOWERY SAVINGS BANK CARD LAYOUT – INTERNAL USE

APPLICATION _____ PAGE _____ OF _____

PROCEDURE OR RUN NO. _____ FORM NO. _____

NOTES OR INSTRUCTIONS _____

VOLUME: AVG. _____ MAX. _____ FREQUENCY _____



SOURCE	DATA TYPE	FIELDS	COLUMNS

Fig. 3-4. Card Layout Form. (*Courtesy*, The Bowery Savings Bank)

MULTIPLE CARD FORM
INPUT PARAMETER CONTROL CARDS

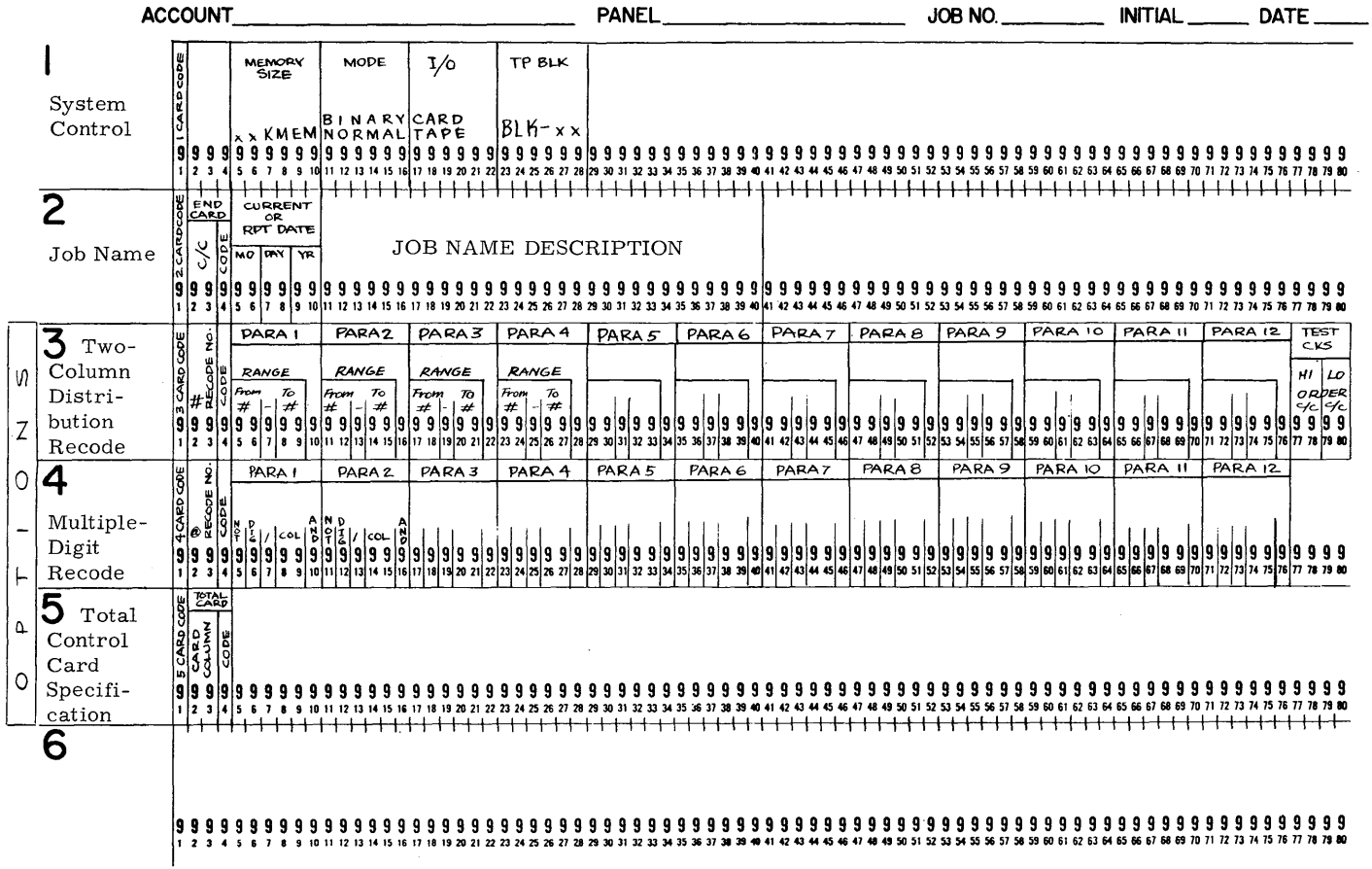


Fig. 3-5. Multiple Card Layout Form. (*Courtesy*, The Diebold Group, Inc.)

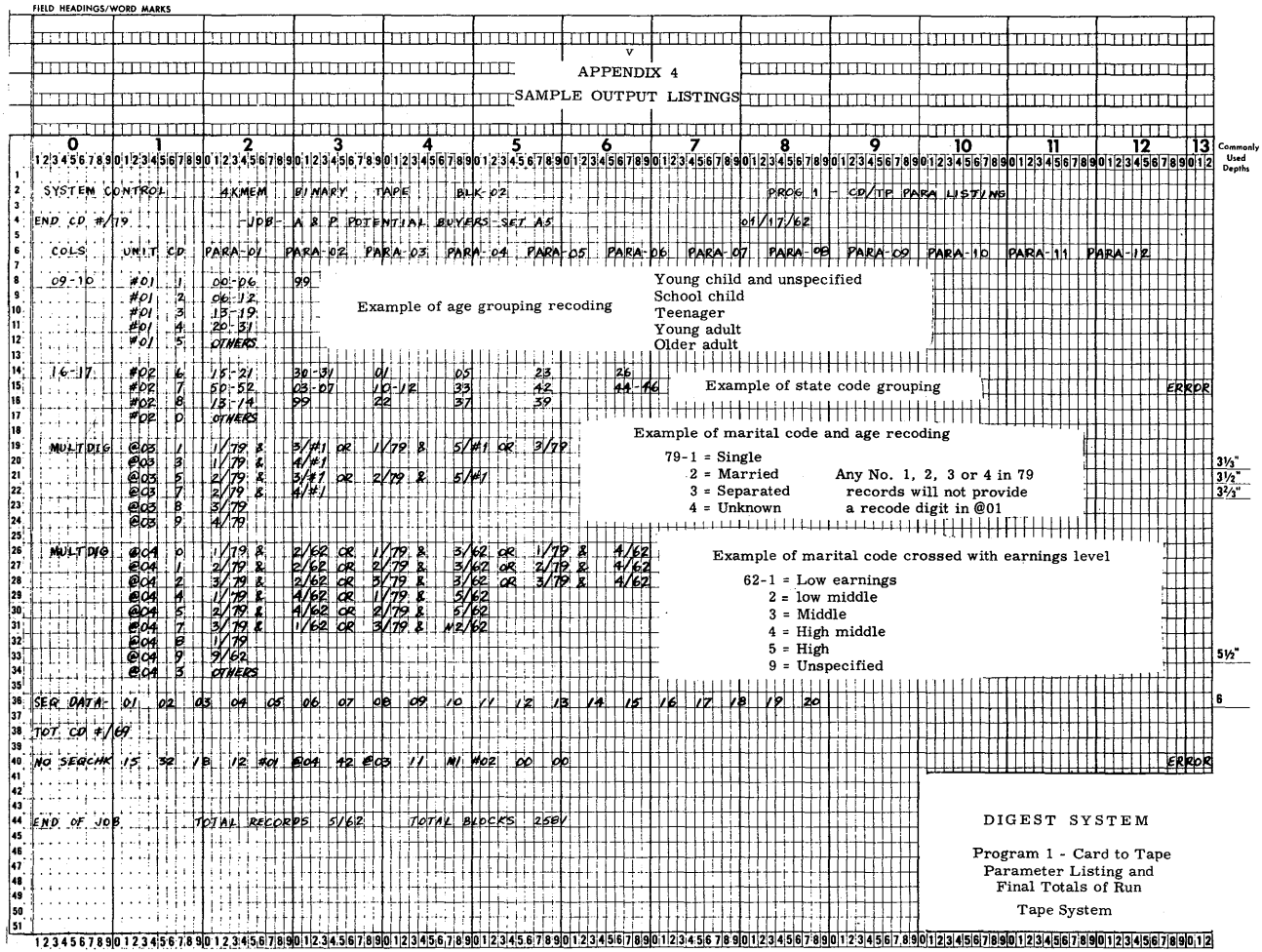


Fig. 3-6. Printer Layout Form. (*Courtesy*, The Diebold Group, Inc.)

Printer Layout Standards.—Layout forms are provided by equipment manufacturers or by manufacturers of paper forms. Such layouts should be drawn carefully, using the following rules:

1. A printer layout form shall be drawn for each report prepared.
2. If the report is printed on a custom form or a stock imprint, a sample of the form or a page proof shall be included as part of the required layout.
3. The printer layout will be drawn on standard form No. *XXX*.
4. Two complete lines of printing shall be shown for each different format line possible; the first line will show the limits of the fields by having continuous records for each field. The second line will have an actual sample of the information that is represented.
5. Above each two-line set the field names shall be indicated. The field names should be standard, and should correspond to the names used on all other layouts and to the names used in the standard coding system. Where abbreviations are used they should be separately referenced in a legend page which accompanies the block diagram.*
6. If headings are printed, they should constitute the first set of lines on the layout sheet.
7. Special descriptive notes and explanations which are not a definite part of the layout (i.e., they will not be printed by the machine) should be drawn or typed in a style separate from the printing used for the layout.
8. The source of the field shall be indicated directly below; in this manner a field can be taken from a card field, a tape record, or calculated.

A sample printer layout is included as Figure 3-6.

Tape Layout Standards.—The layout of tape records is more difficult than the layout of cards or printer records because tape records are variable in length. Also, since tapes are an auxiliary storage medium, they are frequently used in many different program runs and their layout is common to all. Standards are suggested below:

1. A tape layout record shall be prepared for each file used in the system.
2. A tape layout record shall be drawn on standard form No. *XXX*.
3. The tape layout record will include:

   The system name
   The tape record name
   A list of programs in which the tape is used
   The name of the program in which the file is created
   A layout of the fields including
      Field name
      Field mnemonic or abbreviation
      Field length
      Data type
      Location of decimal points or other pertinent information

* Also see Chapter IV.

4. A cross-reference book will be maintained showing the tape files used in each program, and the programs in which each tape file is used. This cross reference will be used to maintain control of all changes made to tape record layouts.

All equipment manufacturers supply tape layout forms. If desired a special form may be designed. Tape layout forms are illustrated as Figures 3-7*a* and 3-7*b*.

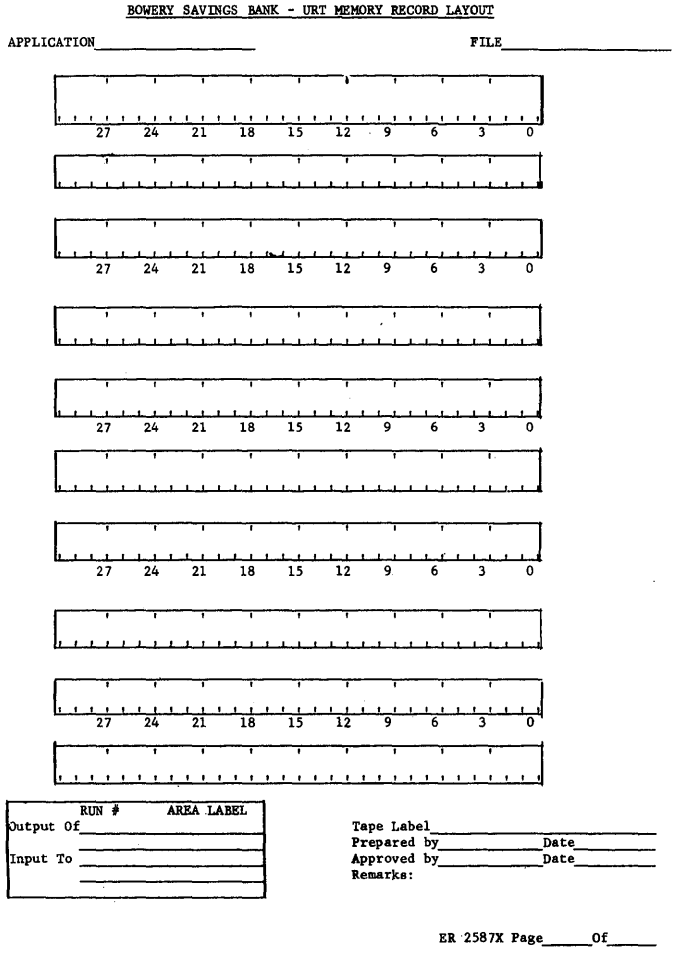


Fig. 3-7*a*. Memory Record Layout Form. (*Courtesy*, The Bowery Savings Bank)

Memory Layout Standards.—The programmer, after finishing his program, is responsible for providing a layout of memory utilization as a

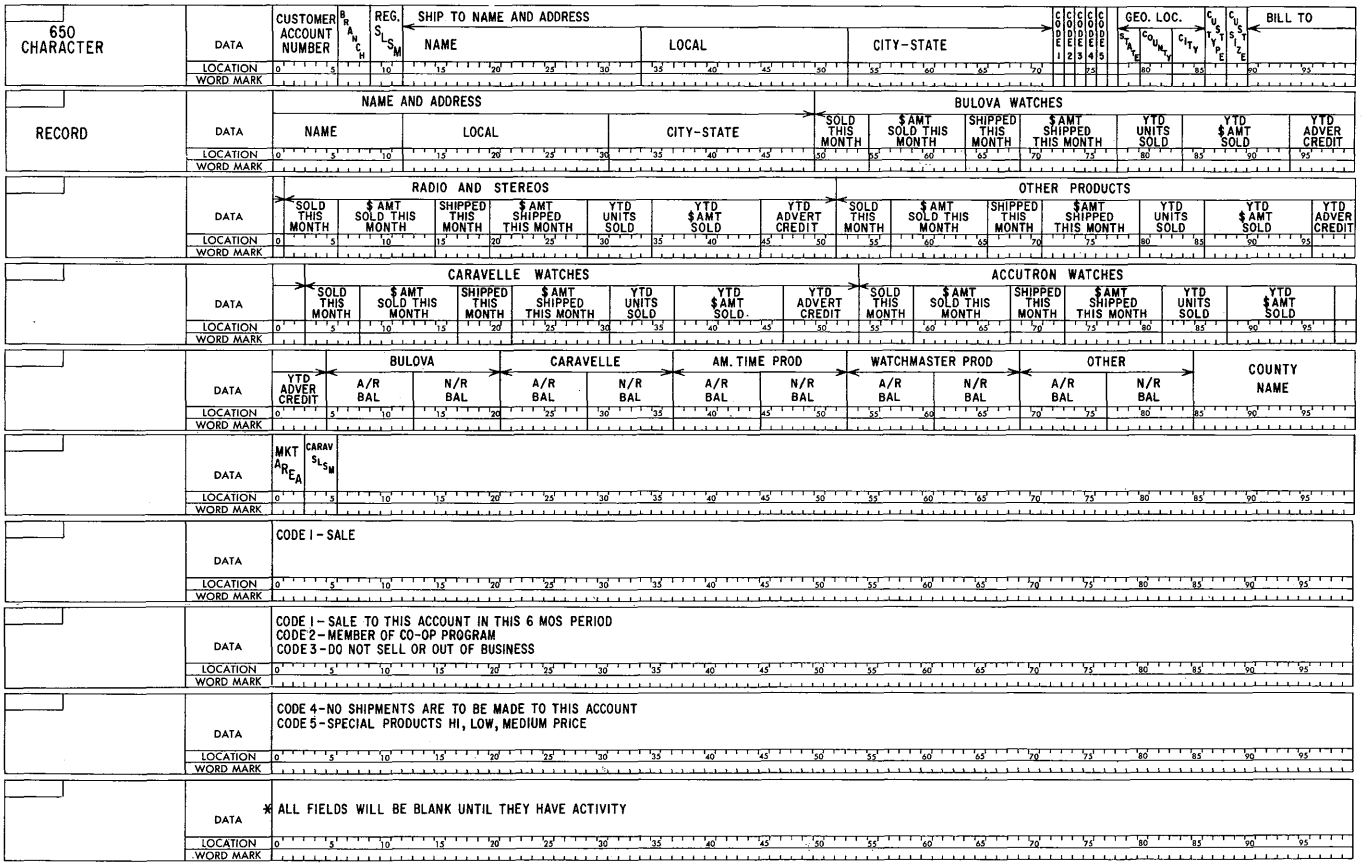


Fig. 3-7*b*. Tape Layout Form. (*Courtesy*, Bulova Watch Company)

part of the standard documentation. This is discussed in more detail in Chapter V. A standard memory layout form is shown as Figure 3-8. Occasionally the systems analyst is required to make up a layout of a section of memory when he is designating the use of specially constructed tables of one form or another. In this case the standard layout form for tapes can be adapted as a memory layout. It may also be possible to obtain a separate memory layout form from the manufacturer, especially if memory divides logically into segments or modules of a specific size.

Other layouts that may be required include record layouts on an auxiliary drum, or on a disc or card random access file. For all of these, a basic tape record layout form which divides the total record into segments of 100 characters or words will be quite adequate. If the installation requires many drum or disc layouts, special standards can be developed along lines similar to the rules shown for tape record layouts.

Document and Procedure Analysis

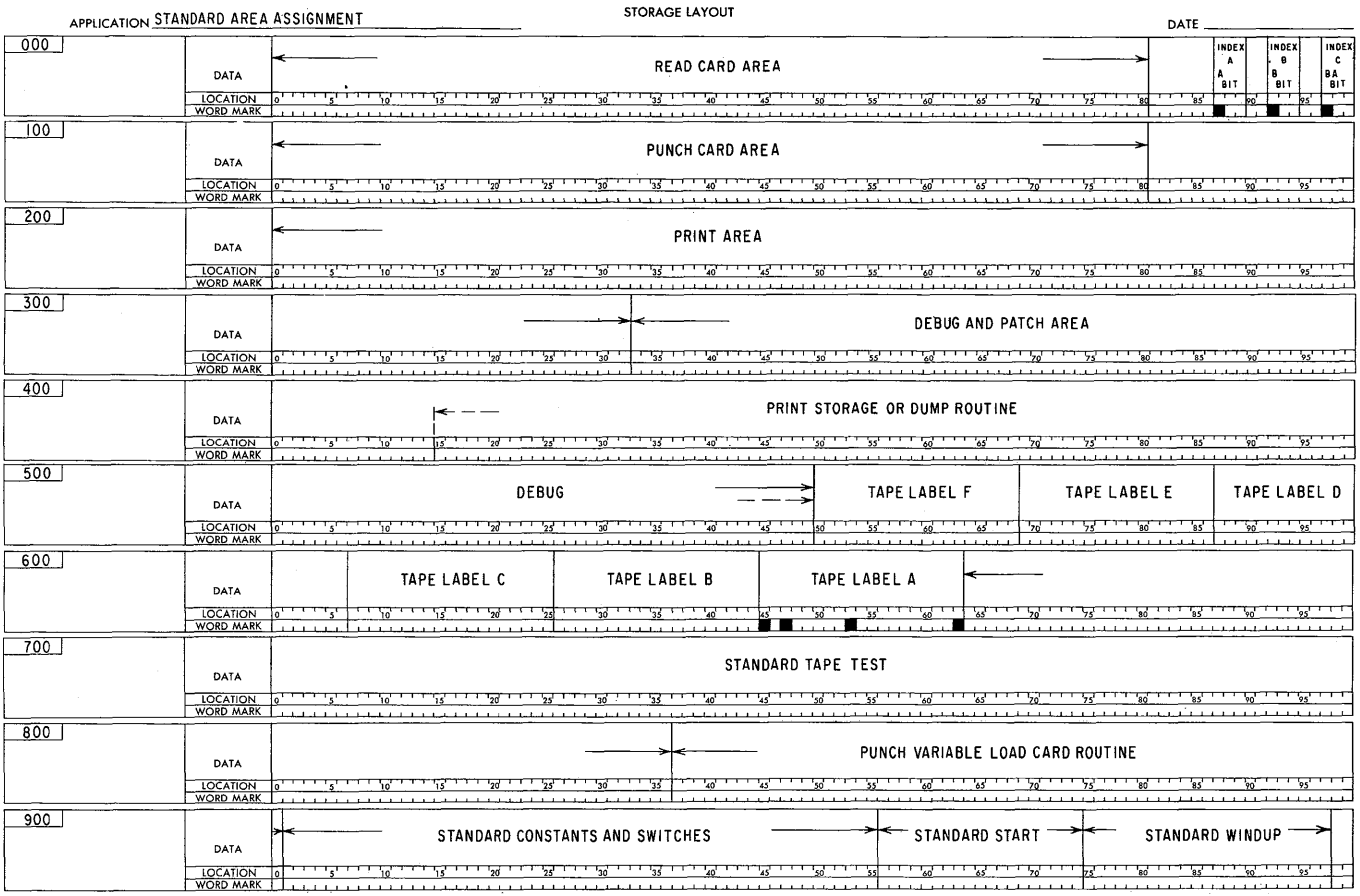
One of the more difficult functions of systems analysis is the review and evaluation of existing documents and procedures. It is almost always necessary to completely evaluate existing procedures and their outputs before an adequate new system can be designed. A standard procedure and a standard methodology should be provided to enable effective procedure analysis.

Procedure Analysis.—The analysis of existing procedures is a function generally reserved for the problem analyst. Nonetheless, the systems analyst must have an understanding of analytical techniques, which are very similar to the techniques of computer systems analysis. These standards are appropriate:

1. For every procedure analyzed a flowchart shall be constructed showing the flow of information through its operations. On the flowchart, each step of the procedure shall be assigned one symbol and shall be numbered consecutively. The process shall be described in writing alongside the flowchart or on a continuing page, in sequence by step number.
2. As a part of the flowchart, documents shall be briefly described and referenced by form number. For each document an estimate shall be made and recorded of the volumes used in a standard period of time.

A sample flowchart form is illustrated in Figure 3-9.

Document Analysis.—Each procedure is made up of a flowchart which shows the flow of information through one or more departments. This information generally flows in document form, i.e., there are standard forms used to transmit the information from one place to the next. It is necessary to analyze each of these documents to determine whether



FOLLOWING THE STANDARD ROUTINE IS AN ORG-1000 FOR THE DEFINITION PAGE. THE PROGRAMMER SHOULD NEXT ASSIGN IN ORDER: READ ROUTINES WITH THEIR CONTROLS, WRITE CONTROLS, WRITE ROUTINES WITH THEIR CONTROLS, LABELS, TAPE AREAS, WORK AREAS AND ANY OTHER AREAS THE PROGRAMMER MAY WISH TO ASSIGN. THE PROGRAMMER SHOULD THEN ASSIGN THE ORIGIN OF THE PROGRAM AT THE NEXT AVAILABLE LOCATION.

Fig. 3-8. Storage Layout.

POST OFFICE DEPARTMENT

PROCEDURE ANALYSIS

DOC. NO.	DOCUMENT DESCRIPTION	VOLUME	FLOW DIAGRAM	STEP	PROCESS	VOLUME	EQUIP.	TIME
	INPUT							
	OUTPUT							
REMARKS								

INDEX NO. | PROCEDURE TITLE | PERIOD | DATE | ANALYST | PAGE

Fig. 3-9. Procedure Analysis Form. (*Courtesy*, The United States Post Office Department)

- The document is really necessary
- The information is not available in another form
- The proposed system may eliminate the transfer of this information
- The document can be simplified, combined, made smaller or less frequent, to reduce overall costs.

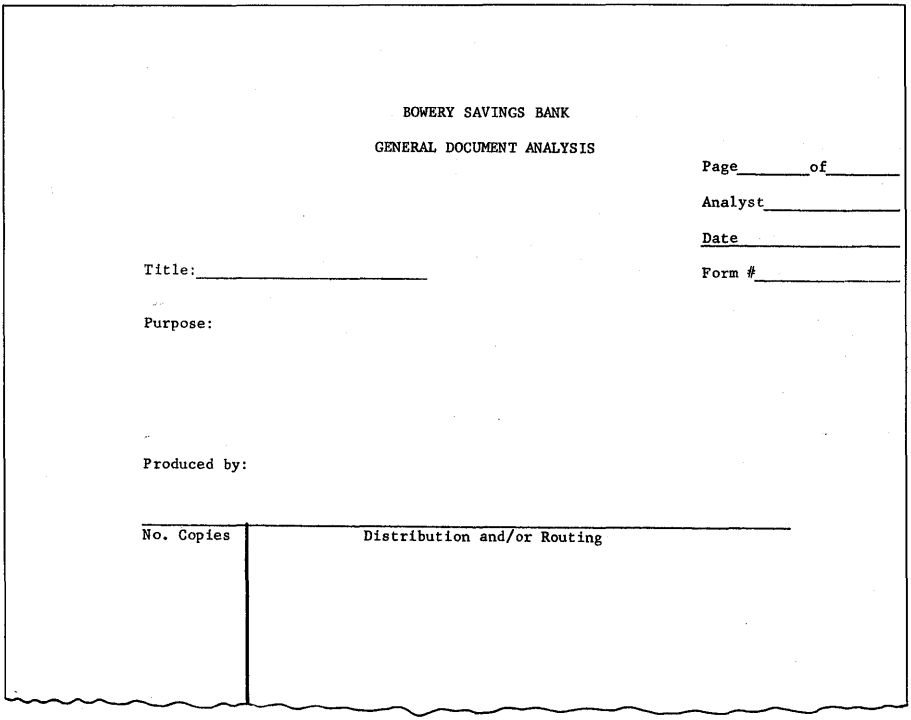
To standardize the presentation of document analysis, the following rules are suggested:

1. For each document described on the flowchart, a document analysis shall be made.
2. The document analysis shall consist of
   - A general document description form
   - An annotated copy of the pertinent document
   - A detailed analysis of the document and its disposition
3. The general document description form shall indicate
   - Name and number of the document
   - Number of copies
   - Routing and distribution
   - Purpose of the document
   - Medium and method of preparation

Two such general document descriptions are shown in Figure 3-10*a* and 3-10*b*.

4. The detailed document description may be incorporated as a part of the general description or may be given on a separate form. The detailed analysis will reference the annotated copy of the form, where each distinct field of information has been separately and uniquely numbered.
5. The detailed document analysis will show
   - Document name, number, and major distribution
   - Periodic volume, size, and the number and distribution of copies
   - Users, and the fields in which the user is interested
   - Frequency of use or referral by each user
   - Whether or not the same information is available in another form or format, with the same or better frequency
6. The detailed document analysis will further show, for each field:
   - Title of the field and its reference to the annotated form
   - Size of the field and its characteristics
   - Position of the field; i.e., where it is located in relation to the margin
   - Line number and associated spacing
   - Source of the field
   - Contribution of the field to further calculations or to other fields

A sample form which can be used for detailed document analysis is shown in Figure 3-11.



BOWERY SAVINGS BANK

GENERAL DOCUMENT ANALYSIS

Page_____of_____

Analyst_____

Date_____

Title:_____ Form #_____

Purpose:

Produced by:

No. Copies	Distribution and/or Routing

Fig. 3-10*a*. General Document Analysis. (*Courtesy*, The Bowery Savings Bank)

Problem Definition

The most difficult link in the communications process is the manner in which the problem is to be defined. There is a great need for a "systems discipline," but until the method of the defining task is changed, it will be difficult to achieve. One approach is the detailed document analysis that has been illustrated. A much more advanced approach has been developed and implemented by a major insurance company with which the author has been associated as a consultant. This approach has been termed the "notation" system; it uses a complete lexicon of mathematical notation to rigidly define each element of the problem, and its associated solution. This approach is extremely effective but it requires considerable training on the part of the user.

A third approach to the problem has been suggested by John W. Young, Jr. and Henry K. Kent of the National Cash Register Company.*

* John W. Young, Jr. and Henry K. Kent, "Abstract Formulation of Data Processing Problems," November-December, 1958, *Journal of Industrial Engineering*, a publication of the American Institute of Industrial Engineers. (With permission of the copyright holders.)

POST OFFICE DEPARTMENT

DOCUMENT DESCRIPTION

DOCUMENT NAME			DOCUMENT NO.	
OTHER NAMES USED			LAYOUT NO.	
			FORM NO.	
AUTHORITY	PERIOD		NO. OF COPIES	VOLUME
MEDIA	HOW PREPARED			
OPERATIONS INVOLVED IN				
REMARKS				

CONTENTS

NO.	DATA NAME	FREQUENCY	CHARACTERS	A/N	ORIGIN

DATE ANALYST SOURCE PAGE

POD Form 1273
Feb. 1962

STUDY

POD, WASH., D. C.

Fig. 3-10*b*. Document Description. (*Courtesy* The United States Post Office Department)

This approach is related to the "notation" system, in that it provides for a complete set of mathematical symbols to define each element of a given problem. The approach goes further in using a graphical method of describing the relationships between the input fields and the output information. The entire system can be described with one diagram.

BOWERY SAVINGS BANK

DETAILED DOCUMENT ANALYSIS

Page_____of_____

Analyst_____

Date_____

Form #_____

Title:_____

Estimated Volume:_____Per_____ Input ☐ Output ☐

Width:_____Length:_____ No. of Copies_____

Special Conditions:_____

Entry No.	Title and/or Explanation	% Freq. or Wght.	No. of Characters	A/N	Wghtd. No. of Bits	Col. # or Print Position	Line No.	Spacing

Fig. 3-11. Detailed Document Analysis. (*Courtesy*, The Bowery Savings Bank)

This method, and its documentation is illustrated as Figure 3-12*a, b, c, d.* The illustration shows that the authors have defined four components of a data processing problem:

- Information sets, described with the letter *P*
- Documents, described with the letter *D*
- Relationships
- Operational requirements, or parameters of the problem.

Information sets are merely the data which make up the problem. In the illustration the information set consists of a number of items which describe an invoicing system. The first item is the date (P_1), the second is the customer number (P_2), and so forth.

Documents are, of course, the basic outputs of any system. Documents are subscripted in a random sequence, so that the invoice is given the number D_2. Each distinct field is given a specific notation, so that a field can be fully defined with two subscripts. The date on the invoice is therefore D_{2-1}, the first field of the second document.

Relationships are expressed in strict mathematical notation. The date D_{2-1} is the information set P_1. Since it consists of month, day, and year,

which in themselves are distinct and useful data items, the date $P_1 = P_7 \times P_8 \times P_9$. Operational requirements, the last component in the problem, define the external parameters of the problem. The volume or average number of D_2 is defined as D_2/P_7E (Real time), or invoices per day. Other operational requirements can be similarly expressed, using the symbol table defined in the illustration.

Control Coding Standards

One of the first items to be standardized is the numbering scheme to be used for programs, systems, forms, tapes and other files. A relatively simple system can be established, as follows:

1. All systems or applications shall be assigned a system letter in sequence.
2. All programs shall be numbered within the system letter, consecutively in order of assignment. The program number shall carry the frequency code as a distinct part of the number.

Therefore

AO1W (System A, Program 1, run Weekly)
AO2D (System A, Program 2, run Daily)

3. Report numbers shall be assigned on the basis of the program number which creates the report. Reports shall be numbered consecutively within the system, and shall be followed by the system letter and program number in which they are created.

001 AO1W is the first report of System A, created by program 01
007 AO3D is the seventh report of System A, created by program 03

4. Tape file numbers shall be assigned consecutively, and shall be preceded by the run in which they are created. Thus, A01W001 is the tape file number for file 001, created in program A01W.

The use of arbitrary numbering systems is not recommended. It is far easier to develop a cohesive standard numbering system to cover all aspects of the data processing program, thus eliminating operator confusion. A sample standards manual page that illustrates this concept is shown as Figure 3-13.

Flowcharting Standards

For desired uniformity of flowcharts, the analyst must be given standards on kind of flowcharts, format, method of preparation, and the information to be shown, as shown below.

Kind of Flowcharts to be Prepared

1. For every application there shall be *one* flowchart which shows the overall flow of information through the computer operation. This is referred to as the macro-flowchart or "big-picture" chart.

Information Sets

		n	L	Relationships
P_1	Date	—	$6N$	$P_1 = P_7 \times P_8 \times P_9$
P_2	Customer Identification No.	2000	$5N$	$P_2 \cong P_{10}$, $P_2 \sim P_4$
P_3	Ship to code	9	$1N$	$P_{14} \cong P_2 \times P_3$
P_4	Salesman No.	50	$2N$	$P_2 \sim P_4 \sim P_{11}$
P_5	Model No.	150	$5A/N$	$P_5 \cong P_{12} \times P_{13}$
P_6	Quantity ordered	—	$2N$	
P_7	Day	31	$2N$	$P_1 = P_7 \times P_8 \times P_9$
P_8	Month	12	$2A$	$P_1 = P_7 \times P_8 \times P_9$
P_9	Year	10	$2N$	$P_1 = P_7 \times P_8 \times P_9$
P_{10}	Customer N/A	2000	$50A/N$	$P_{10} \cong P_2$
P_{11}	Warehouse name	10	$12A$	$P_4 \sim P_{11} \sim P_{15}$
P_{12}	Part No.	800	$3A/N$	$P_5 \cong P_{12} \times P_{13}$
P_{13}	Color	20	$2A$	$P_5 \cong P_{12} \times P_{13}$
P_{14}	Ship to address	6000	$50A/N$	$P_{14} \cong P_2 \times P_3$
P_{15}	Pricing area	8	$1A$	$P_5 \times P_{15} \sim P_{17}$
P_{16}	Invoice No. (Shipping Notice No.)	—	$5N$	$P_{16} \cong D_2$
P_{17}	Unit price	—	$5N$	$P_5 \times P_{15} \sim P_{17}$
P_{18}	Salesman name	50	$15A$	$P_{15} \cong P_4$

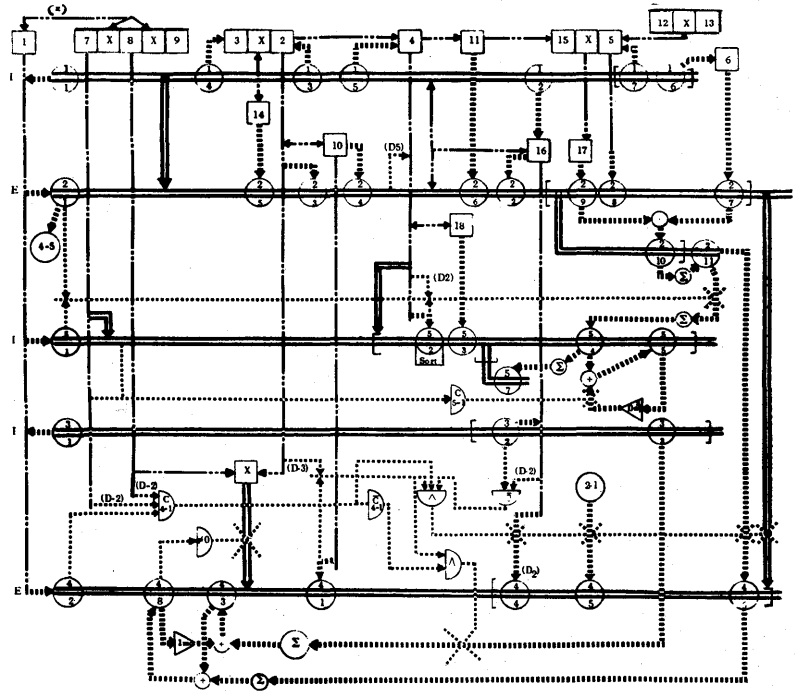
n = number of elements in set
L = number of characters (numeric—N, alphabetic—A, or alphanumeric A/N) in each element.

a. Table 1

List of Symbols

P_i	A list of all possible information belonging to the same class
p_i	A specific member of the class
D_j	A document
d_j	A specific document belonging to the D_j class
D_{jk}	A collection of entries on the document D_j
[ ]	Line item
$\cong$	Isomorphic (one to one correspondence)
$\sim$	Homomorphic (many to one correspondence)
$\times$	Cartesian product, e.g., $P_j \times P_k$ means a pair of p_i and p_k
$\subset$	Contained in
$\rightarrow$	Produces
E	Extrinsic time (real time)
I	Intrinsic time, e.g., date written on document
( )	Function
C_{m-n}	nth Condition relating to the mth Document
|	If
$\overline{C}_{m-n}$	Negation of C_{m-n}, i.e., $\overline{C}_{m-n}$ is true if and only C_{m-n} is false
$\sqsubset$	Number of
$\sqsubset\!\!\sqsubset$	Average number of
$\exists$	There exists
$\nexists$	There does not exist

b. Table 2



d. Complete Graphical Representation

Fig. 3-12. Abstract Notation. (*From* John Young and Henry Kent: "Abstract Formulation of Data Processing Problems," American Institute of Industrial Engineers, Journal of Industrial Engineering, November-December 1958. *Courtesy*, Authors and Publisher)

Document Descriptions

Shipping Notice—D1—Input

Items	Verbal Description	Information Set	Defining Relationship
D_{1-1}	Date	P_1	
D_{1-2}	Shipping Notice No.	P_{16}	
D_{1-3}	Customer Identification No.	P_2	
D_{1-4}	Ship to Code	P_3	
D_{1-5}	Salesman No.	P_4	
[D_{1-6}	Quantity of Order	P_6	
D_{1-7}]	Model No.	P_5	
D_{1-8}	Line Item		$D_{1-6,7}$

Volume: $\overline{\subset} D_1/P_{7E}=300$
$\overline{\subset} D_{1-8}/D_1=5$

Invoice—D2—Output

Items	Verbal Description	Information Set	Defining Relationship
D_{2-1}	Date	P_1	$t_E(D_2)$
D_{2-2}	Invoice No.	P_{16}	
D_{2-3}	Customer Identification No.	P_2	
D_{2-4}	Customer Name and Address	P_{10}	
D_{2-5}	Ship to Address	P_{14}	
D_{2-6}	Warehouse shipped from	P_{11}	
[D_{2-7}	Quantity of order	P_6	
D_{2-8}	Model No.	P_5	
D_{2-9}	Unit Price	P_{17}	$P_{17}(D_{2-8}\times P_{15})$
D_{2-10}]	Extended Price		$D_{2-7}\cdot D_{2-9}$
D_{2-11}	Total Price		$\Sigma\, D_{2-10}$
D_{2-12}	Line Item		$D_{2-7\text{ to }10}$

Producing Relationship: $D_1\rightarrow D_2$
Operational Requirement
Volume: $\overline{\subset} D_2/P_{7E}=300$
Time: $t_E(D_2)-t_E(D_1)<2$ days

Customer Payment—D3—Input

Items	Verbal Description	Information Set	Defining Relationship
D_{3-1}	Date	P_1	
[D_{3-2}	Invoice No.	P_{16}	
D_{3-3}]	Amount		
D_{3-4}	Line Item		$D_{3-2,3}$

Volume: $\overline{\subset} D_3/D_{7E}=200$
$\overline{\subset} D_{3-4}/D_3=1.5$

Monthy Statement—D4—Output

Items	Verbal Description	Information Set	Defining Relationship
D_{4-1}	Customer Name and Address	P_{10}	
D_{4-2}	Date	P_1	$10/P_{3E}(D_4)/P_{3E}(D_4)$ Statements are to be dated the 10th of the month.
D_{4-3}	Customer's (old) balance		$D_{4-3}(D_{4-1},\ P_3-1)-\Sigma\, \overline{C}_{4-1}D_{3-3}(D_{4-1})$
[D_{4-4}	Invoice No.	P_{16}	D_{2-2}
D_{4-5}	Date of Invoice	P_1	D_{2-1}
D_{4-6}]	Amount of Invoice		D_{2-11}
D_{4-7}	Line Item		$D_{4-4,5,6}$
D_{4-8}	New Balance		$D_{4-3}+\Sigma\, D_{4-6}$

Producing Relationship: $P_2\times P_3\rightarrow D_4 \mid D_{4-3}\neq 0$
(statements are produced each month for each customer with a non-zero balance)
$D_2\rightarrow D_{4-7} \mid C_{4-1}\wedge C_{4-2}$
(an invoice is included in the statement if both condition C_{4-1} and C_{4-2} are true)

Special Conditions: C_{4-1}: $[P_3(D_2)=P_3(D_4)\wedge P_7(D_2)<10]\vee[P_3(D_4)-1=P_3(D_3)\wedge P_7(D_2)>10]$
(the invoice was dated after the 10th of the preceding month but before the 10th of this month.)
C_{4-2}: $\overline{\exists} D_3[D_{3-2}(D_{2-2})]$
(a payment has not been received for the invoice)

Operational Requirements:
Volume: $\overline{\subset} D_4/P_{3E}=500$
(the average number of statements issued per month is 500)
$\overline{\subset} D_{4-7}/D_4=4$
(the average number of invoices (line items) itemized per statement is 4)
Time: $10<P_{7E}(D_4)<15$
(statements are to be produced between the 10th and the 15th of the month)

Daily Cumulative Sales Report—D5—Output

Items	Verbal Descriptions	Information Set	Defining Relationship
D_{5-1}	Date	P_1	
[D_{5-2}	Salesman No.	P_4	
D_{5-3}	Salesman Name	P_{18}	
D_{5-4}	Sales this date		$\Sigma\, D_{2-11}(D_{5-1},\ D_{5-2})$
D_{5-5}]	Cumulative sales this month		$D_{5-4}+C_{5-1}\cdot D_{5-5}(D_{5-1}-1)$
D_{5-6}	Line Item		$D_{5-2,3,4,5}$
D_{5-7}	Total gross sales this date		$\Sigma\, D_{5-4}$

Producing Relationships: $P_1\rightarrow D_5$
$P_4\rightarrow D_{5-6}$
Conditions: C_{5-1}: $P_7(D_{5-1})\neq 1$
Operational Requirements:
Volume: $\subset D_{5-6}/D_5=50$
Time: $t_E(D_5)-t_1(D_5)<2$ days

c. Table 3

F. CONTROL CODING STANDARDS

Numbering of Programs

1. The format of Program Number is as follows:

System Letter	Run No.	Frequency Code
←1→	← 2 →	←1→

2. The System Letter is assigned as follows:

REAL - Real Time Savings
D - Deposit Accounting
M - Mortgage Accounting
P - Payroll
C - Christmas Club
U - Utility, Generalized or "Canned" Subroutine
B - Bond Accounting
G - General Expense Accounting
S - Safe Deposit
E - Executive Routines

3. The Run Number is assigned consecutively based upon the order in which executed, in order of frequency.

4. The frequency code is a letter denoting the frequency of operation as follows:

C - Conversion
D - Daily
W - Weekly
B - Bi-Weekly
M - Monthly
Q - Quarterly
S - Semi-Annually
A - Annually
R - Upon Request

Numbering of Files

5. The format of File Number is as follows:

System Letter	Run No.	Frequency	File No.
← 1 →	← 2 →	← 1 →	← 3 →

6. The system letter, run no. and frequency are assigned from the run that creates the file.

7. The file number is assigned consecutively for each application.

Fig. 3-13. Control Coding Standards. (*Courtesy*, The Bowery Savings Bank)

2. The purpose of the macro-flowchart is to provide a complete understanding of the information flow; it will show the linkage between all programs, the source of the input and the disposition of the output. It will be used as a master chart for the entire application, and will be made available to all concerned.

3. The macro-flowchart shall be drawn on one continuous page, if at all possible. The symbols used on the macro-flowchart shall be the same as the symbols used for the program flowchart.

4. A program flowchart will be drawn for each and every program used in the system, including standard programs such as sorts and merges.

5. The program flowchart shall be drawn on standard form No. *XXX*, 8½ x 11″, and shall be a part of the primary documentation of the program.

6. The program flowchart (micro-flowchart) shall include:
   - An input-output diagram of the machine components used
   - A statement of the functions of the program
   - A statement of the special features and options of the program
   - An identifying block showing the program name
     - the program number
     - the analyst
     - the date
     - the application
     - the total operating time
   - A summary of machine run time estimates
   - A detailed analysis of machine run time estimates

Method of Preparing the Flowchart.—As indicated in Figures 3-14 and 3-15, a standard form should be used to prepare the flowchart for each program. Included in the flowchart should be a clear statement of the functions of the program, derived from the macro-chart which showed the major purpose of the run. A separate timing analysis should be made in order to estimate theoretical run time, using the average expected volumes. This is necessary to insure against major errors in the program. A serious flaw in the logic will be indicated if actual run time shows an increase of 50% or more over the original estimate with no increase in volume.

Two exhibits illustrate different methods of developing system logic. The first, Figure 3-14, represents a tape and card merge that creates a number of outputs. It clearly segregates set-up time from running time, primarily because the computer for which it was developed is rented, and rent is suspended during set-up. The second, Figure 3-15, gives less emphasis to timing and more to the program functions. The equipment is purchased, operating a real-time program in parallel with the object program. Timing is less critical and set-up time is treated like operating time.

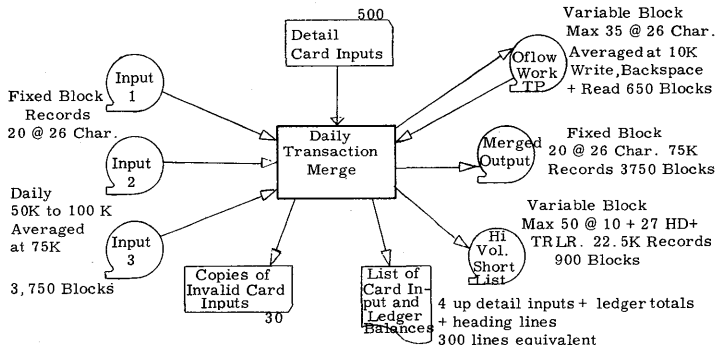
Flowchart symbols differ from those used for block diagrams, which show each program processing step. The manufacturer usually supplies a template carrying both flowcharting and block diagramming symbols. A clear distinction must be made between the two, and a clear definition provided for each symbol. The following standards may be applied:

1. The flowchart shall be drawn using only the symbols shown below to describe the inputs and outputs.

PROGRAM FUNCTIONS

1. Merge 3 Input tape files and card Inputs to produce the merged output tape
2. Add Ledger code to each record; on a change in ledger, print out ledger totals and write tape control records.
3. Total all check detail records to provide a "sight PAY" check control record. Write this header record in front of check details on the merged output.
4. Validity check the card input and list four up. If invalid, punch out and omit from the merged output.
5. On high volume accounts, substitute package post records for the details on the merged output, and write the detail check records on the short list tape.

ACCT. NAME State Street Bank NO. 76
JOB NAME Demand Deposit NO.
PROG NAME Daily Transaction Merge
PROGRAMMER C. Brennan
DATE April 1962
PAGE 1 OF 1
TOTAL TIME 30 (17)



SUMMARY OF MACHINE TIME

	CARD READING	CARD PUNCHING	PRINTING	TAPE	PROCESS	INITIAL SET UP	TOTAL
RUNNING	.6	.14	.5	4.2	7.7		13.14
ADD'T'L SET UP			3.0	12.0		2.0	17.0
TOTAL	.6	.14	3.5	16.2	7.7	2.0	30.14

CARD READING — VOL. 500; TIME 75MS — .6 minutes

CARD PUNCHING — VOL. 30; TIME 250 MS — .14 minutes

PRINTING — REPORT: Journal; NUMBER OF LINES: 300 @ EQV LV 100 MS; ADD'L SET UP TIME: 3 minutes — .5 minute

S.U. 3.0 minutes

TAPE

	Inputs 1,2,3	Merged Output	Short List	Oflow WKTP Write	Oflow WKTP Read	
NAME OR NUMBER						
NUMBER OF RECORDS	75K @ AV	75K @ AV.	22.5 K @	10K @	10K @	@
	.97 min.	.97 min.	.2 min.	.023 min	.037 min	
REWIND	2 min.	- OVER LAPPED		EOJ - REWIND		
REEL CHANGE						

4.2 minutes

S.U. 12. minutes

PROCESS 75K ACTIVE RECORDS @ 6.164 MS PER SEC. TOT 7.7 minutes

INACTIVE RECORDS @ PER SEC. TOT

7.7 minutes

INITIAL SET UP

PRINTER	TAPE UNITS	ALL OTHER
3 minutes	12 minutes	2 minutes

S.U. 17 minutes

TOTAL 30.14 minutes

TOTAL S.U. 17 minutes

Fig. 3-14. Flowchart and Flowchart Timing Analysis. (*Courtesy*, The State Street Bank and Trust Company *and* The Diebold Group, Inc.)

BOWERY SAVINGS BANK

II. URT PROCESS CHART

TIMING ESTIMATES	
Punch 5M cards	35 Min.
Print 2M lines	Overlap
Read Tape	Overlap
Set Up	5 Min.
TOTAL	40 Min.

RUN NAME Inactive Acct. Search RUN NO. D01A

PROGRAMMER Hutt PAGE OF

DATE PREPARED 15 Apr. '62 REV # 2

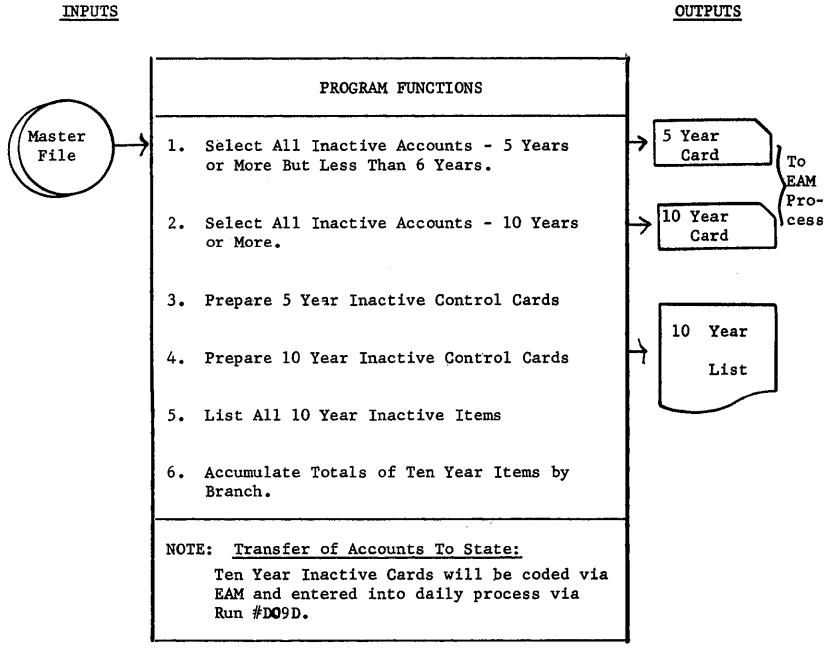


Fig. 3-15. Flowchart. (*Courtesy*, The Bowery Savings Bank)

Figures 3-16 and 3-17 illustrate two different sets of symbols. The pages are taken from the standards manuals of two companies; the symbology is strictly enforced.

2. Each input or output symbol shall be accompanied by the following information:
   - Record size (if not standard)
   - Record volume
   - Blocking factor
   - Data description or type, i.e., binary, fieldata, etc.
   - Name of file

3. The central computer symbol shall be accompanied by the following:
   - Name and number of run
   - Frequency of running
   - Machine used

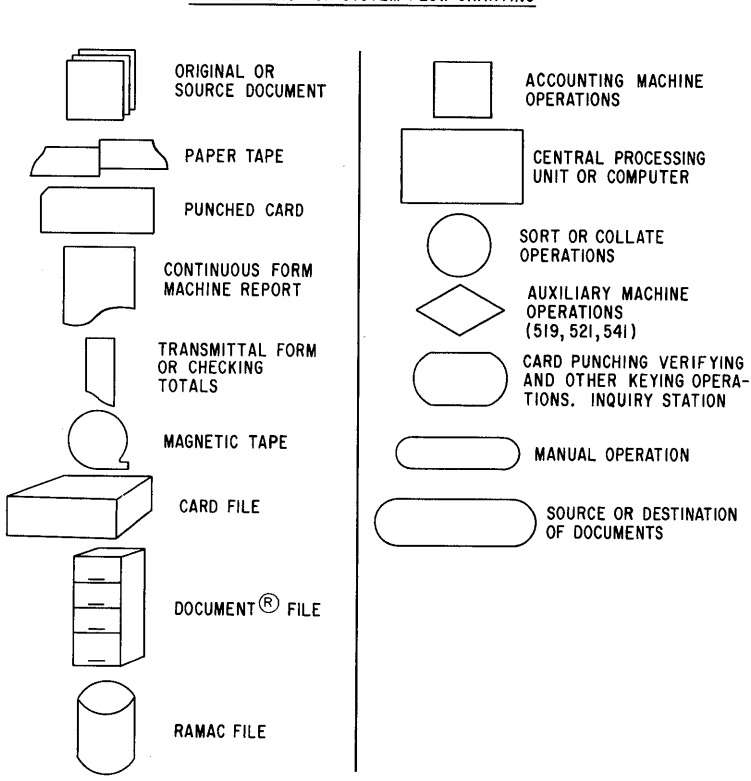


Fig. 3-16. Flowcharting Symbols. (*Courtesy*, The Bulova Watch Company)

The Job Specification Manual

The basic output of systems analysis is a complete description of the task to be performed, complete with layouts and flowcharts. This is the "job specification manual." Standards for preparation and use of the job specification manual assist in overcoming communications difficulties among users, analysts and programmers. These may include:

1. The systems analysts shall prepare a complete job specification manual for each computer application.
2. The purposes of the job specification manual are to:
   —document and describe the system
   —explain system outputs and functions
   —state system requirements for programmers
   —avoid misunderstandings among the involved departments
3. The completed job specification manual shall be approved in writing by the managers of the using departments.

SECTION I - PROGRAMMING

A. PROCESS CHARTING CONVENTIONS

1. A process chart must be prepared for each program. This chart will also serve as part of the primary documentation of the program.

2. Process charts will be drawn on form no. ER 2584X, Process Chart Illustration. (IA-3)

3. The purpose of the process chart is to show the flow of input and output through an individual run.

4. The following conventions are to be used:

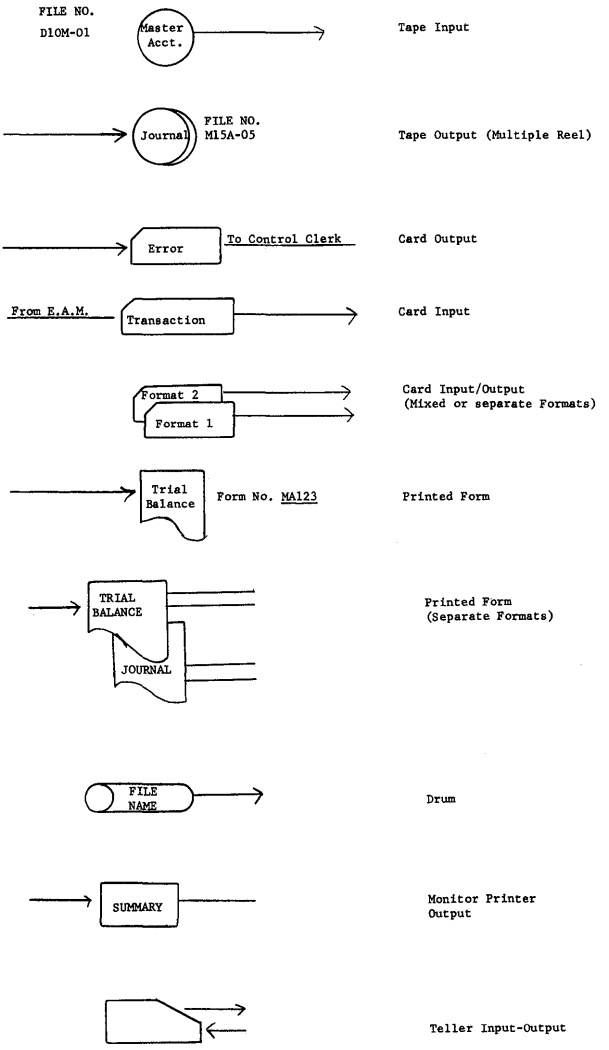


Fig. 3-17. Flowcharting Symbols. (*Courtesy*, The Bowery Savings Bank)

4. The job specification manual, after such approval, shall be transmitted to the programming department.

5. The programming department shall review the specification within seven working days for normal jobs or two working days for "crash" jobs. If accepted, further responsibility for implementation and accuracy rests wholly with the programming department.

6. The programming department may reject the job specification for reasons of incompleteness, lack of clarity, or failure to define requisite elements. The reasons for rejection must be given in writing to the systems analyst.

7. Upon acceptance the programming department shall prepare a schedule and an estimate of costs and time for implementation of the job.

8. The following shall be contained in the Job Specification Manual:

Introductory Pages:

Title Pages Showing the System Name and Letter
Table of Contents
Revision Page

Scope of the System—The first section defines the scope of the system, and indicates which departments supply information to/and receive information from the system.

General Description of the Existing System—The existing system is generally described and its functions, purposes and method of operation concisely outlined.

Flow of the Existing System—A detailed procedure analysis of the existing system shall include a complete flowchart.

Outputs of the Existing System—The documents produced by the existing system are listed and briefly described, including distribution and use made of each.

General Description of the New System—The proposed new system is generally described, including a statement of its purposes and functions and the major differences from the existing system. A brief statement of the reasons for and advantages of change should also be included.

Flow of the New System—An overall flowchart of the new system is included. This flowchart graphically shows the flow of the system from and to the computer operation and provides a verbal description of the flow within the computer department.

Output Layouts—The outputs of the new system are described and a detailed layout provided for each of the output documents.

Output Distribution—The distribution of the new output documents is indicated and the number of copies, routing and purpose in each department shown. The output distribution is further summarized to show what each department will receive as a part of the proposed system.

Input Layouts—The inputs of the new system are described, and complete layouts of the input documents and input cards or tapes will be provided.

Input Responsibility—The source of each input document is indicated, and the department or user responsible for each item on the input documents. This information is to be summarized by department.

Macro-Logic—The overall logic of the internal flow will be briefly described by the systems analyst, wherever useful. If a problem in understanding is sure to result from a complex logical point, the systems analyst may actually draw the macro-logic chart for that function.

Files to be Maintained—The specification will contain a listing of the

tape, card or other permanent record files to be maintained, and the items of information to be included with each file. There will also be complete layouts of the files which require updating. There need not be complete layouts for intermediate or work files; these may be prepared later by the programmer.

List of Programs—A detailed schedule of the programs to be written shall be a part of the systems specification. For each program it shall include:

Suggested name of the program
Run frequency
Inputs and outputs
Documents to be created
Files to be maintained

Timing Estimates—A summary of approximate computer timing is provided by the systems analyst, based on the volumes expected for each of the system inputs and outputs. Timing for each run should be multiplied by its annual frequency and divided to provide approximate daily machine time cost.

Controls—Wherever controls are established or eliminated, the specification must fully describe each one. This shall include type of control, and the method in which it will be balanced.

Problem Definition—The problem definition is a detailed narrative or an abstract notation. In either case it must contain the description of all relationships and requirements.

Audit Trail—A separate section of the system specification shows the audit trail of all financial information. It indicates the methods with which errors and defalcations will be prevented or eliminated through balance controls.

Glossary—The developed glossary of terms peculiar to the system is the first or the last part of the job specification manual.

9. Whenever a question occurs in relation to the intent of the analyst in the detail of the specification the programmer shall contact the analyst directly. A small revision in the specification may sometimes markedly reduce the programming effort or computer operating time. In these instances the programmer and the analyst will work together to create the optimal systems design.

CHAPTER SUMMARY

The major functions of the systems analyst are the development of a computer-oriented system through a detailed analysis of the existing system. The ultimate output of the analysis is a detailed job specification, containing all of the tools necessary to produce a series of computer programs. Standards must be established in the systems analysis area to reduce the communications problems between the analyst, the programmer and the using department. These standards include the method and symbology to be used in drawing layouts and flowcharts, assigning number codes to program elements, and analyzing the system's documents. This chapter develops some basic standards in these areas, and has concluded with the standards which specify the contents of the job specification manual.

Questions for Review

1. Describe the functions of the systems analyst and the programmer.
2. What type of layouts are required in systems analysis?
3. What is the contents of the job specification manual?
4. Why is the job specification necessary?
5. Why is it necessary to perform document analysis?
6. What type of glossary is necessary in systems design?
7. Develop a glossary of terms peculiar to your own business.

Chapter IV

METHODS STANDARDS: PROGRAMMING, PART 1

THE SCOPE OF PROGRAMMING

After the systems analyst has developed and obtained approval of the final job specification manual, it is submitted to the programmer for translation to machine instructions. The programmer performs the following tasks:

- Logical analysis—translation of the program functions into a block diagram to provide a graphic representation of the steps the machine will follow.
- Coding—translation of the diagram into a symbolic language.
- Desk checking—a detailed review of the program steps.
- Test data preparation—the creation of a set of input data to verify that all conditions have been properly recognized in the program.
- Assembly and test—machine translation of the symbolic language into machine language and actual program operation under test conditions.
- Documentation—the preparation of a description of the program and its operation.
- Installation—assistance to the operating department in conversion and parallel operation, and correcting any errors that occur.

For each of these tasks rigid discipline is necessary for

- A uniform product
- Efficient scheduling
- Performance evaluation
- Interchangeability of personnel

Programming lends itself extremely well to the establishment of a rigid discipline. Programming is a highly methodical task requiring attention to the most minute detail. Standards are practical and can be easily installed.

Since programming comprises more tasks than systems analysis and operation, two chapters have been devoted to it. This chapter discusses all tasks up to and including the coding function. Chapter V deals with standards and rules for all other elements.

USING THE JOB SPECIFICATION MANUAL

The Job Specification Manual was developed by the systems analyst as the complete description and documentation for the entire operation. The programmer must recognize that the job specification is his only input; there should be no further reason for him to contact the user. If there are any questions or inadequacy in the specification they should be resolved by the systems analyst. Accordingly, there should be some simple rules set down to assist the programmer in reviewing the specification and in determining adequacy, such as the following:

1. The Job Specification Manual is to be reviewed in detail for completeness and accuracy. The manual should contain a complete systems description, and a set of flowcharts of all the programs which make up the system. If the manual is incomplete or unclear or not sufficiently specific the programmer may reject the manual, stating in writing the exact reasons for this action.
2. If the manual is considered sufficiently descriptive, the programmer must accept the specification; he must realize that complete responsibility for the implementation of the system will be his after acceptance.
3. Those parts of the manual of major interest to the programmer are:
   - Scope of the manual
   - General description of the new system
   - Flowcharts of the new system
   - Output layouts
   - Input layouts
   - Macro-logic, where applicable
   - Files to be maintained
   - List of programs, their schedule and the timing estimates
   - Controls to be built into the programs
   - Problem definition
   - Glossary
4. The programmer must first review the systems design: the overall flowchart which describes the system's operation. This design should be reviewed for accuracy, for conformity to installation standards, and to be sure that it properly optimizes computer use.
5. The programmer's second task is to develop a macro-block diagram for each program.*

* The techniques for block diagramming are outlined in the next section.

6. After completion of macro-block diagrams, the programmer will review the logic in detail with the systems analyst. This review has the following purposes:

To insure that the programmer properly understands the requirements
To develop the best and most comprehensive overall logic
To assist the analyst in determining whether or not the system is optimal

7. The systems analyst will approve the macro-logic after this review. All further changes to the program must be approved by the analyst, the programmer and their respective supervisors.

STANDARDS FOR LOGICAL ANALYSIS

The drawing of block diagrams can become individualistic and arbitrary. To avoid this, a detailed set of rules must define:

- Kind of diagrams to be drawn
- Format
- Method of diagramming
- Complexity or level of detail
- Coding scheme

Rules for Block Diagrams

Basic definitions, such as the following, should be included:

1. Two kinds of block diagrams shall be prepared for each program. The first depicts the major logic of the program and is generally referred to as a "macro" block diagram. The second kind is more detailed, showing the logic of each of the major program operations. This is the "micro" block diagram, often called "semi-detailed" block diagram. (It is not necessary to show one box or symbol for each machine instruction; if that were done, no diagram would be necessary.)
2. The purpose of the macro block diagram is to show all the major elements that make up the program and their relationships. The purpose of the micro block diagram is to show the logical sequence of all decisions, data movement, calculations, and linkages which fulfill the objective of the program. Both diagrams should be machine-independent; i.e., the program logic shown should be capable of translation into the language of any machine having a similar configuration.
3. The relationship between the macro and the micro diagram should be clearly maintained; each symbol on the macro diagram should correspond directly to the micro diagram on which the detail is shown.

Format Rules

Certain rules of format are necessary, even though they appear rudimentary. Do not allow assumptions to be made; state the requirements:

4. Paper shall be white bond, 8½" x 11". A margin of 1" shall be maintained on the left side for binding and reasonable margins shall be observed on all other sides.

5. All block diagrams shall carry an identification block on the upper right corner of the page. This block will show the following minimum information:

Program Number
Program Name
Programmer
Date and Revision Number
Block Number and Block Description

6. The page number and the total number of pages (page *X* of *Y*) shall be clearly noted on the bottom right of each page of the diagram.

Method of Diagramming

Although the manufacturer-supplied template (as illustrated in Figure 4-1) almost forces a uniform method of diagramming, basic rules and symbol definitions must be included as part of the standards manual. Variations should not be allowed, despite the status attached to ownership of a ten-year old "original" template:

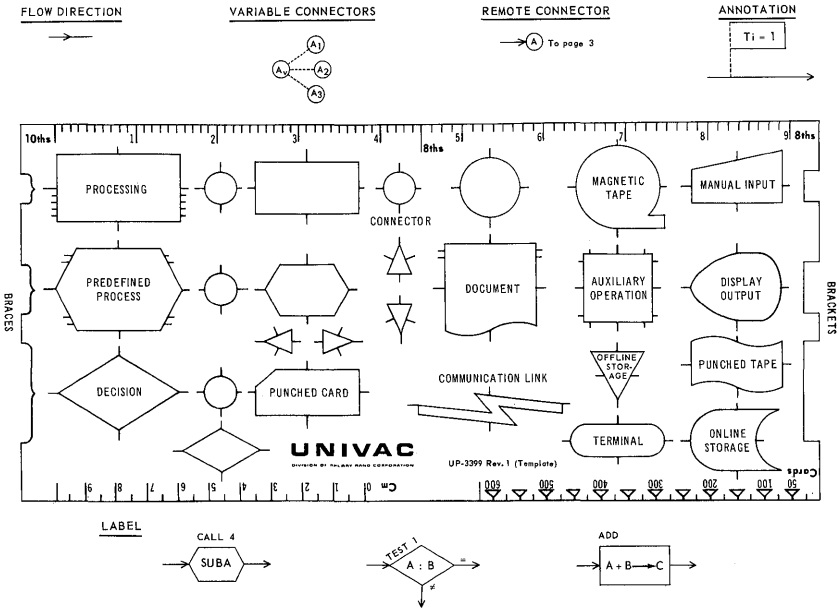


Fig. 4-1. Charting and Diagramming Template. (*Courtesy*, UNIVAC Division of Sperry Rand Corporation)

7. The operations shown should be concise and self-explanatory. Whenever symbols or abbreviations are used they should conform to the established standards. Nonstandard abbreviations should be separately defined on a legend page to precede the block diagrams.

The conventions to be used may differ from one installation to another, but should be absolutely uniform within any installation. Figure 4-2 and Figure 4-3 depict two different conventions.

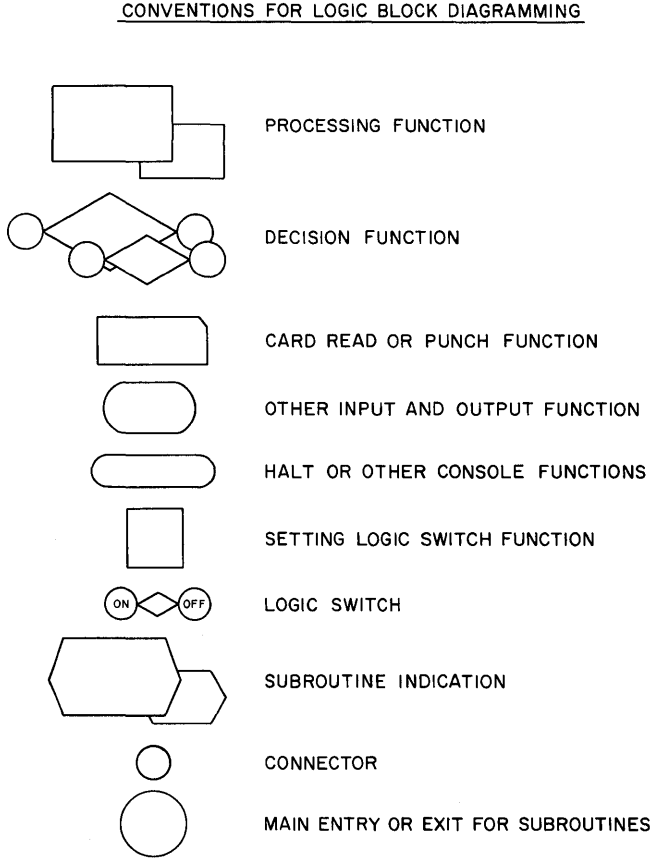


Fig. 4-2. Block Diagramming Symbols. (*Courtesy*, Bulova Watch Company)

Complexity

The most difficult standard to establish is the degree of block diagram detail. The meticulous programmer often overdraws, thus creating a machine dependent diagram with one symbol for each instruction. In this case no diagram is really necessary since the coding serves this

B. FLOWCHARTING CONVENTIONS

1. For each program, two flowcharts are to be prepared. The first will be a "macro-chart", showing overall logic only. The second will be a "micro chart" showing detailed logic.

2. Rule for macro flowcharts

   a. A macro flowchart should show the major operations of a program in the form of linked subroutines or program segments.

   b. Conventions for macro flowcharts are defined below (Section 8).

   c. Each symbol on the macro flowchart should be labeled and correspond directly to the micro flowchart which shows the detail.

3. All flowcharts shall be imprinted with an identification block, (stamp), showing programmer, date, approvals, etc.

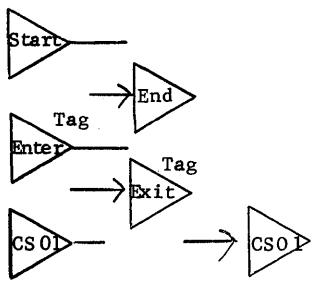
4. The purpose of a micro flowchart is to show in logical sequence the detailed decisions, movements of data and computations that are necessary to achieve the desired results. A flow chart should express the tasks performed, not the machine instructions used to accomplish these tasks. The flowchart should be separate from the method of implementation.

5. All operations shown should be self-explanatory and should require no additional notes. English statements should be used wherever possible. When symbols and abbreviations are used, a legend page should be inserted as the first page of the flowcharts.

6. If a program uses a subroutine that has been completely documented elsewhere, the micro-flowchart of the subroutine should be omitted. Only the name of the subroutine should be indicated. (This applies only to subroutines documented as standard subroutine packages.)

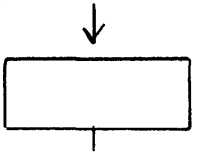
7. Flowcharts should be related to the coding using standard labels as defined in section ID. A complete illustration of the procedure will be found in the sample program.

8. The following conventions are to be used on macro and micro flow diagrams:



Indicate the start and end (where the end is not a computer stop) of a program or the entry and exits points of a subroutine. For a program or run, the triangle should contain the words "Start" and "End". For a Library Subroutine, the symbol should contain the name of the subroutine. For a subroutine within the program being documented, the symbol should contain the words "Entry" and "Exit" for each entry and exit. In all cases a tag or appropriate notation should be shown above the symbol to facilitate cross reference to coding. Both symbols should point in direction of flow.

Fig. 4-3. Flowcharting Conventions. (*Courtesy*, The Bowery Savings Bank)

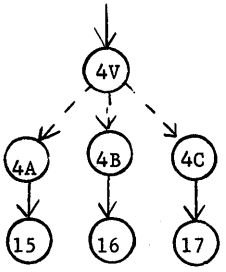


OPERATION BOX

One operation box will be included for each logical task to be accomplished by the program (except as noted below).

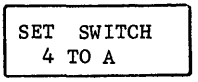
CONNECTOR CIRCLE

A connector circle indicates the juncture of two or more program branches. The numeric which is associated with the label should be inserted in the circle. The number assigned to a connector will be the same as the "to" location, step number. (i.e. the box following the return point). If connectors are on different pages of the chart, TO and FROM page numbers should be written beside the affected connectors.



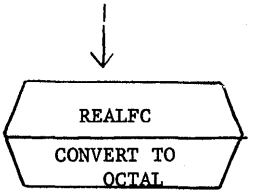
VARIABLE CONNECTOR

Variable connectors, or multi-exit decision switches must be assigned numeric notations in sequence with normal connectors. Do not re-use numbers that have already been assigned within the same subroutine. A "V" subscript is used in the common entry point to the variable switch. Alpha subscripts, starting with "A" should be assigned to each exit point and related to the jump point as shown.



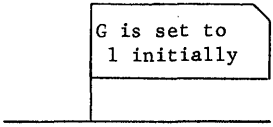
Set Variable Connector. Indicates the setting of a switch to be used later in the program.

Indicates the direction of flow. Flow paths in general should be from left to right and from top to bottom of the page, but this rule may be violated to avoid the use of too many connectors, or to allow for less complicated chart design.

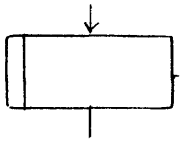


Reference to another program segment or a closed subroutine. The execution of a separately defined closed subroutine at a point in the flow. This symbol implies further definition of the subroutine, in the documentation of the run of which it is a part. The base label of the subroutine or segment and its name should be written within the symbol.

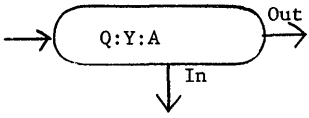
Fig. 4-3. (cont.)



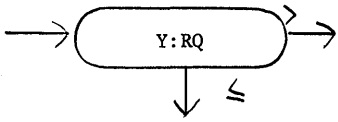
Assertion: Used to assert or explain the existence of certain conditions at some point in the flow path. Upper right hand corner should be slanted.



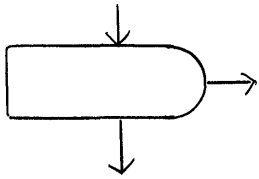
Substitution box. Refers to a counting operation such as the advancing of a counter or incrementing of an index register.



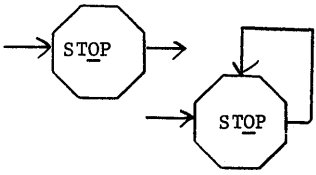
Comparison Test - 3 Way
Q = Contents of Register Q
Y = Quantity being compared
A = Contents of Register A
The failing side of the comparison should follow the right hand path.



Comparison Test - 2 Way
RQ = Contents of Register Q
Y = Quantity being compared to Q.
The failing side of the comparison should follow the right hand path.

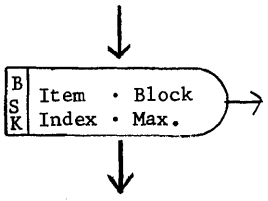


Multiple Operation involving both operation and a decision.

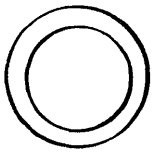


Computer Stop: Indicates those points in the program where the computer comes to a stop and which require manual intervention for restarting. The word "STOP" should be enclosed in the octagon.

Fig. 4-3. (cont.)



Multiple Operation involving changing an Index register and a decision (B skip or B jump instructions)



Library Subroutine

The execution of an open or closed library subroutine at a point in the flow. The identification of the library routine should be written within the symbol. The detailed flow chart for the library routine should not appear in the overall chart for the routine calline on the library.

Fig. 4-3. (cont.)

purpose. The less meticulous worker will frequently insert a symbol stating "calculate" or "update" without proper regard to the many detailed steps that make up the calculation or the updating. It is necessary to control the level of diagramming. Some typical rules are given below:

8. The program shall be divided into logical segments or subroutines, referred to as *blocks*. Each program shall have no more than *26* such blocks, and no fewer than six.
9. A program block shall be subdivided into logical functions called *steps*. Each program block shall have no more than *99* steps and no fewer than ten.

The above rules serve to define the limits of complexity. Dependent on the nature of the programs and the size of the machine, the limits may be changed to fit the circumstances.

10. A macro-block diagram shall be limited to one page. It uses one symbol for each of the possible 26 blocks, showing all links between the blocks.
11. A micro-block diagram shall consist of not more than two pages for each block, and only one block may be shown on a page.

Coding Scheme

In order to provide appropriate linkage and permit cross reference among the macro-diagram, the micro diagram and the coding, a meaningful coding scheme must be developed. The following scheme has

been designed to avoid the use of "input connectors" and to provide a direct and automatic connection to the coding. It uses the standard labeling system discussed in the latter part of this Chapter.

12. Each block will be assigned a letter, in sequence, as related to the normal program flow. The first block should always be assigned the letter A and the sequence maintained.

13. Each symbol or step within a block shall be assigned a number from 01 to 99 continuously. The general sequence is from top to bottom of the micro block diagram page but exact numerical order is not required. The number is written outside of the symbol, on the upper righthand corner. Later insertions and changes may be given a decimal notation within the same sequence, e.g. 02.1.

14. The macro block diagram shall clearly note the block letter for each of its symbols for which a micro-diagram exists.

15. "Exit connectors" shall be labeled with the block letter and symbol number of the step to which an exit is made. If the *block letter* is not shown, the connection must be made to another symbol within the same block. If the *symbol number* is not shown, the connection must be made to step *X 01*. There is no need to indicate the entry connector, since the block letter and the symbol number appear clearly on the page, as explained in the following rule.

16. The first symbol on each page shall be an entry connector denoting the block letter and symbol number of the first step on this page. This will simplify review of diagrams.

17. Formulas and other special relationships in the logic of the program should be shown on the micro-block diagram on the right margin in a special oversize block. This will emphasize their presence, and the manner in which the depicted logic has been derived.

A set of sample block diagrams are included as Figure 4-4*a* and 4-4*b*. The macro-block diagram shows the major routines of the program, even though it does not indicate the presence of several generalized routines. The micro diagram shows the detailed logic for the generalized tape read routine, which is used in the input tape "get" routines shown as blocks C, D, and E.

CHARACTER WRITING CONVENTIONS

It is quite important that alphabetic and numeric characters are written clearly, so that punch operators can make the distinction between similar characters. Thus, the letter O is frequently confused with the number 0, since the context in which they appear is often meaningless to a punch operator. Since the methods vary from one installation to another and programmers do change jobs, it is important that *one* set of characters is used. There are installations today using the symbols, O, Ø, θ, *0*, Θ, and $\overline{\text{O}}$, to distinguish the letter from the numeral, and other installations use the same symbol for both!

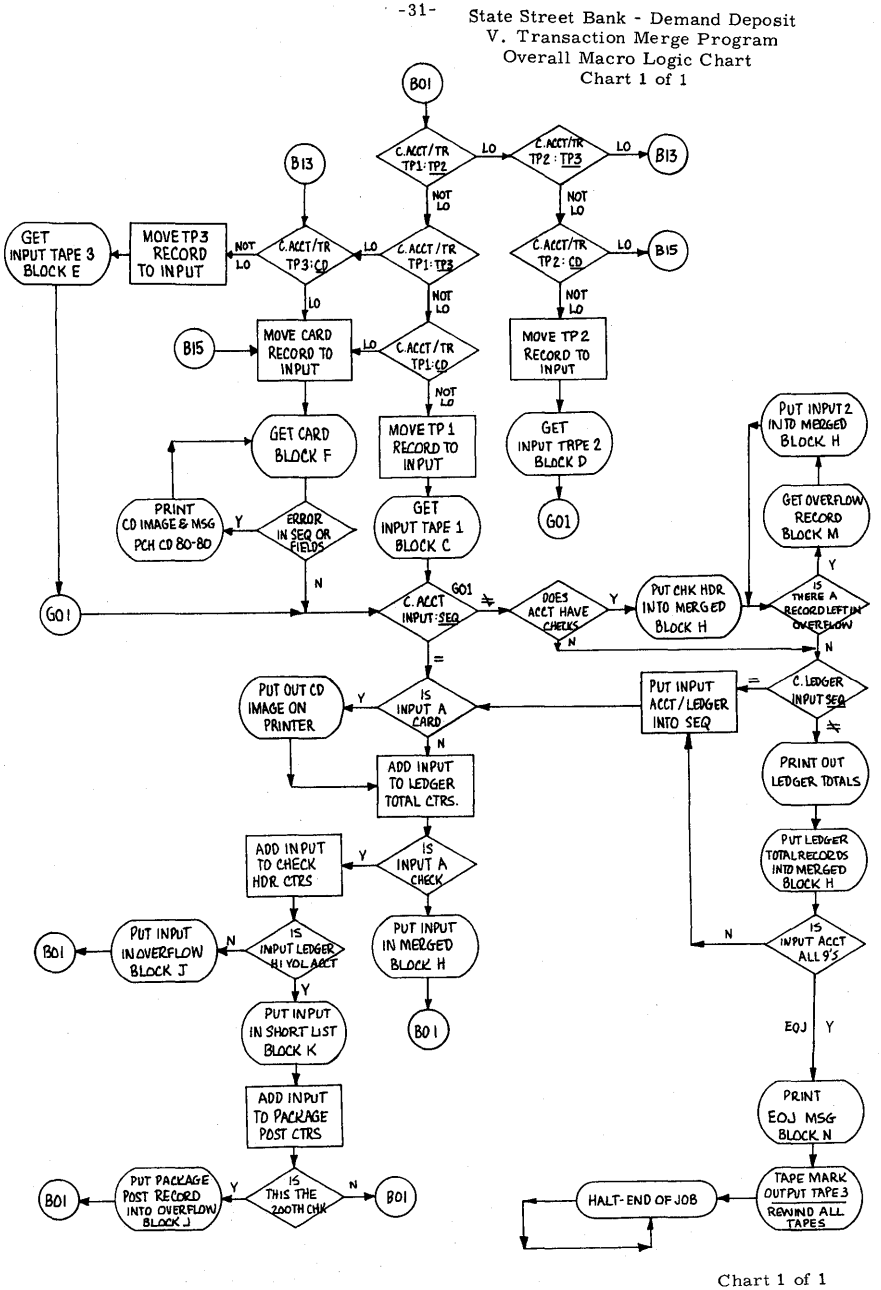


Fig. 4-4*a*. Macro Logic Charts. (*Courtesy*, State Street Bank and Trust Company *and* The Diebold Group, Inc.)

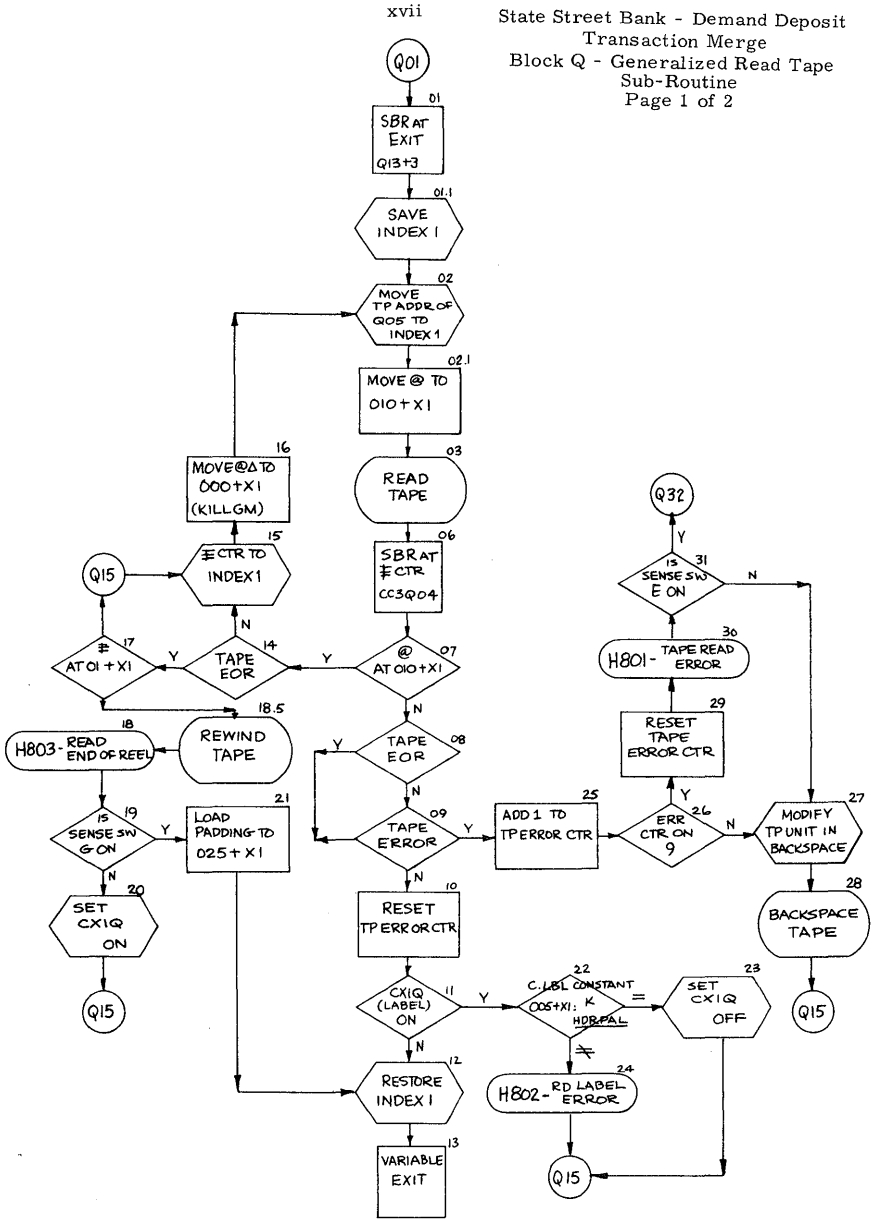


Fig. 4-4*b*. Micro Logic Charts. (*Courtesy*, State Street Bank and Trust Company *and* The Diebold Group, Inc.)

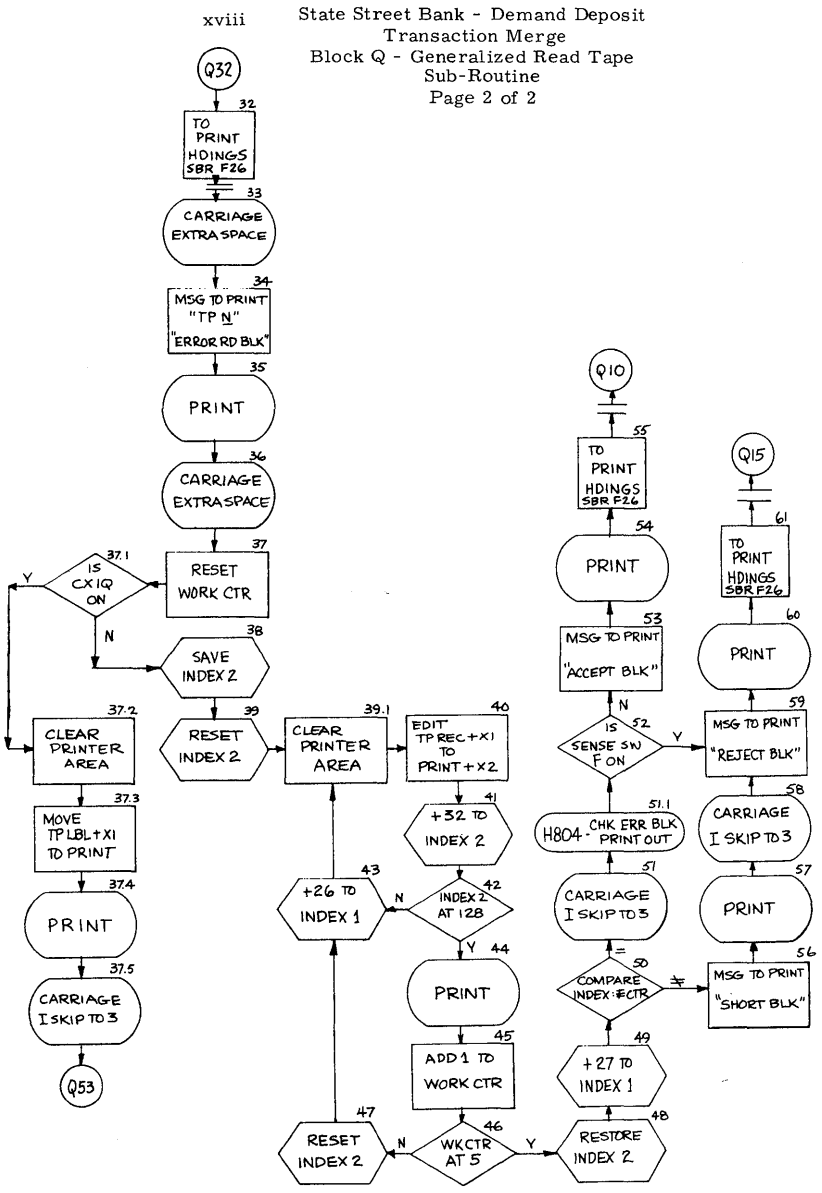


Fig. 4-4*b*. (cont.)

It is similarly useful to set up rules for the writing of special characters that appear in the machine's vocabulary, and for the use of symbols to describe relationships in documentation for block diagramming. Some follow:

1. Characters should be written as follows

The number one: 1
The letter "eye: I

The number zero: 0
The letter "Oh": Ø

The number five: 5
The letter "ess": S

The number two: 2
The letter "zee": Ƶ

The letter "vee": V
The letter "you": U

2. All letters must be written in upper case
3. The following symbols should be used wherever possible to simplify diagrams:

Character	*Description*
b	Blank
:	Compare
>	Greater than
<	Less than
=	Equal
≠	Not equal
≤	Less than or equal to
≥	Greater than or equal to
→	Used within an operation box to signify transfer of data
—	Minus
+	Plus
0	Zero
●	Set
Σ	Sum
C(Y)	Contents of Y
f(x)	Function of x

CODING STANDARDS

After the block diagram has been completed and reviewed, the program is ready to be coded. The coding process translates the logical steps into symbolic computer language. Coding sheets are used whose format corresponds to the required symbolic input form. For all machines the basic elements of coding format are:

- The identification: page and line number, to pinpoint sequence
- The label or tag: a name or number assigned to an instruction or constant for automatic reference

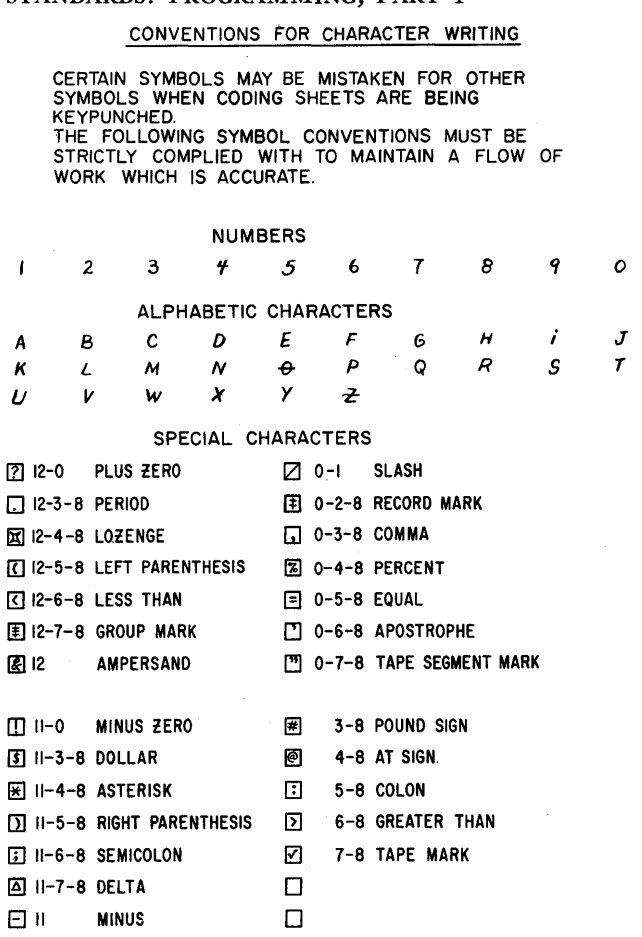


Fig. 4-5. Character Writing Conventions. (*Courtesy*, Bulova Watch Company)

- The operation code: the name or numeric designation of the instruction to be executed
- The operand(s): the address of the data to be operated upon, with increments, indexes, filters or other designations, according to machine characteristics
- Comments: space for explanation and references.

A typical coding format is shown in Figure 4-7, page 96. The best standards for coding include rules for:

- Coding format
- Coding method
- Program organization

Standards established for coding are generally the most significant. The program code listing is used to test, change, and review or segment the program. The code listing must be intelligible to everyone concerned and must therefore be written in a standard manner. The listing must also relate directly to the block diagram, so that standards must include a method for linking the two.

The Use of Mnemonics

Machine coding is oriented to computer design and machine instructions and addresses are usually meaningless combinations of letters and numbers. The instruction: "Reset accumulator 15 and add a factor located in memory at address 34,522 into it" is represented on the IBM 705 as HDEB2. The 1401 instruction MAB2ZYZ means "Execute a data transfer (move) from address 3122 indexed by register 3 to address 5989 indexed by register 1."

A symbolic translator is used to simplify machine coding. The programmer, when writing his coding in the symbolic language, is allowed to use mnenomics to represent instructions and addresses. Reset and Add becomes RAD and the memory location 34,522 might be referenced as NETPAY. The real simplification of the symbolic translator is in allowing the use of *programmer-generated,* pseudo-English phrases which designate data locations and program sections. These phrases are referred to as "tags" or "labels."

The lack of discipline in the use of these labels is a major factor in preventing exchange and understanding between two programmers. One programmer may call his first instructions START, another BEGIN, and still a third by the name of the first function performed, e.g. READ. In large programs, requiring from 400 to 1,000 distinct mnemonic tags, confusion and lack of organization prevail. Misspelling, duplication, lack of sequence, and loss of meaning all contribute to the continued problem; the establishment of a rigorous discipline is the only way in which these problems can be rapidly eliminated.

A further disadvantage of programmer-generated mnemonics is that it becomes extremely difficult to check a large program completely. When examining an instruction such as "ADD CØN1 IPC" it becomes necessary to look through the entire listing to determine the exact value of both CØN1 and IPC. The former could be a CØNstant, with a value of 1; it could also be a constant of size 1, or the first constant in a sequence. IPC could mean Input Counter, or Intermediate Pay Check, or Input Planning Constant, among others.

The burden of assigning distinct names to many program elements frequently forces programmers into a second language, to prevent dupli-

cation. It is not uncommon to find French or German tags, proper names, and even zoological names such as CAT and DØG in a large program. Their lack of meaning further obscures the complexity of the program.

The coding standards developed on the following pages do not permit programmers to generate personal mnemonics. Although this may cause initial resentment among programmers, the expense of a program which cannot be understood by anyone except its author is never justified.

Coding Format

The format of the coding is restricted by the format which the symbolic translator uses. There are certain rules which further assist uniformity. Some are indicated below.

1. The language or translator to be used shall be . . . . . [For example, all 1401 programs shall be assembled using the Symbolic Programming System. Autocoder and the Report Program Generator may not be used unless permission is given by the Data Processing Manager. All 7090 programs shall be written in FØRTRAN. FAP may be used only in cases where FØRTRAN cannot be used, and only with permission of the Manager.]
2. All coding sheets shall carry the programmer name, program name, program number and date of completion.
3. Page numbers shall be assigned in numeric sequence. (An extension of this rule segregates page numbers for certain functions as:

page 00 for Identification and Statement
pages 01 - 10 for Input-Output System
pages 11 - 20 for Housekeeping Functions
pages 21 - 79 for Main Program
pages 80 - 89 for Areas
pages 90 - 99 for Constants)

4. All coding shall be in pencil, using a number 1, 2 or 2½ black lead to facilitate correction. Erasures shall be made completely.
5. All coding shall be double space to permit proper positioning of insertions. The last five lines of every page originally shall be left blank for later insertions.
6. Insertions made at the bottom of the page must be given a line number to correspond to the place where the instruction is to be inserted. In addition, the place where the insertion is to go shall be marked with an arrow in the left hand margin to allow easy review of coding.

Coding Method

To prevent the use of individual mnemonics and to insure readability the framework of standards should be simple to use, yet provide uniformity to the point where two programmers referring to the same point

in the program will unconditionally use the same tag. Similarly, two programmers using the same constant must use the same tag, thereby conserving memory space. Sample rules follow:

7. Comments shall be given and maintained for all lines of coding where appropriate. These comments shall be written in English in the space provided in concise, terse phrases descriptive of the function of the instructions. It is good practice not to spread comment sentences across more than one line; if this is done, subsequent insertions will destroy the meaning.

8. A Comments card shall be inserted at the beginning of each new block. It shall specify the block letter and the block name or function. If the routine is a generalized subroutine, it shall also specify the conditions of entry and exit. [Comments cards use the operator TITLE, SAY, REM, * , etc. to indicate a pseudo-operation which produces a listing line only.]

9. Whenever a major logical change occurs within a block, a Comments card will be used. It may be left blank to generate an extra space on the listing.

10. Whenever it is necessary to use a reference label for a constant or an instruction the standard labeling system outlined below will be used. Variation from the standard labeling system is not permissible except as noted. If it is necessary to use a series of designations for special purposes, the scheme used should be noted both on the program listing and in the documentation.

Standard Labels. The suggested method of standardizing labels represents a new approach to the problem of coding uniformity. The actual standards used depend on the machine, the type of translator, the permissible size of the labels, and the complexity and rigidity desired. The suggested approach for designing a standard labeling system follows these steps:

a. Analyze the labels used in an average program, and determine major classifications. Typical of a major classification breakdown is instruction, constant, area, and index registers. Other categories which could be considered major include program halts, tape label areas, and special tables.

b. Develop a coding scheme for each major category of label used according to the objectives of standardization.

The objectives for *instructions* are:

- A linkage to the block diagram must be indicated.
- A reasonable sequence should be conveyed so that someone following the coding will go forward on the listing for a higher sequence routine and backward for a lower sequence routine. (Using mnemonics, it is impossible to determine the exact location of any tag, unless machine language is used.)
- Instructions may need to be tagged by type or by size (in a variable length machine) or in other significant ways.

The objectives for *constants* are:

- To be able to tell the actual constant used from the tag.
- To be able to recognize the constant type and size (in a variable length machine) or the number of words, bits or half-words (in a fixed-length machine).
- To assist as much as possible in pinpointing violations; e.g. size may be a factor in divide overflow; type may be a factor in arithmetric failure (e.g. an unsigned constant).

The objectives for *areas* are:

- To note the type of area, tape, card, printer, input, output.
- To recognize the field within the area and its relationship to other fields, for possible contiguous transmission.

It is possible to set down similarly the objectives for any type of label for use in a standard labeling system.

c. On the basis of the objectives and the available label size, develop the parameters that will be used to code the label. If the size of the constant is required, for example, two columns of all constant labels may be used for a size code. Similarly, instructions may contain block letter(s) and the step number to provide a linkage to the block diagram.

A complete standard label system, designed in this case for the IBM 1401 computer but generally applicable to other systems, is illustrated below. A summary chart of the system is reproduced as Figure 4-6.

The 1401 Standard Label System

According to a requirements analysis the major breakdown of label types has been made as follows:

- Instructions (branch points)
- Areas
- Constants
- Halts

Six characters are usable in the 1401 Symbolic Programming System. The major label type designation is given one alphabetic character:

Character 1											
A AREA	Char. 2	L - LABEL AREA		T - TAPE AREA		P - PUNCH CARD AREA		R - READ CARD AREA			
	Char. 3	Unique Character to Define The File as Specific File Within Program									
	Char. 4 - 6	Blank for Overall Area Name Mnemonic for Field Name, or Column Number				Blank for Overall Area Name Equivalent Card Column Number					
B BRANCH POINT	Char. 2	Block Letter - From Diagram									
	Char. 3 - 4	Step number - From Diagram Left justified, including leading zero.									
	Char. 5 - 6	Instruction number Left justified, omit leading zero.				USE ONLY IF REQUIRED					
H HALT	Char. 2 - 4	Halt Number (Location of Halt / 4)									
C CONSTANT	Char. 2	L LITERAL	N NUMERIC	C COUNTER	W TEMP WORK AREA	E EDIT WORD	T TABLE	X SWITCH	A ADDRESS CONSTANT	H HEADING	M MESSAGE
	Char. 3	SIZE OF CONSTANT (CODED ALPHA IF EXCESS 9)					Entry sz.	Switch Number	Block Letter	Report Number	Mnemonic Message Name or Contents
	Char. 4	Actual last three significant Characters		Block Letter		Size of Data Field	No of Entries		Step Number		
	Char. 5			Step Number		Number of Decimals	Entry Number				
	Char. 6	No Sign	Sign Control			Block Letter			Instruction No., If Required	Heading Line No. if > 1	
		L	N	*C	*W	*E	T	X	A	H	M

* Elimination of block and/or step results in the establishment of a reusable constant.

Note: Where a tag cannot be properly identified, the use of an X in the 6th position will signal the tag as a rule exception. This exception can only be made if a condition does not fit the standards outlined above.

Fig. 4-6. Summary of Standard Label System.

B for Branch point
A for Areas
C for Constants
H for Halts *

The "B" is used in preference to the I (Instruction) to reduce confusion with the 1. The use of limited alphabetics has these advantages:

- The SPS system requires that all tags start with a letter. The above four letters make it impossible to accidentally use a number.
- It is now possible to use the other 22 letters of the alphabet with complete freedom for standard "closed" subroutines, which can become a part of any program. (If unrestricted tags are used in a program, it becomes difficult to prevent the use of duplicate tags if a standard subroutine is used.)
- It is possible to know immediately what type of operation is being performed. The modification of an instruction will stand out from the accumulation of totals because in one case the Add will be made to a B-tag, and in the other case it will be to a C-tag, or an A-tag.

The remaining five characters of the tag are used in each of these categories as follows:

B—Branch Point.—In order to provide both sequence and linkage to the block diagram, the second character of the label will be the *block letter,* and the third and fourth character will be the *step number.* Thus, the first instruction of most programs is tagged BA01, indicating the first step of block A. A branch to BN24 which occurs in the D block can immediately be traced to the N block, which should follow the M block on the listing. In a few cases it is necessary to use more than one label for the same step. If the step stated, for example, "Calculate FICA Tax" it would be necessary to have several instructions labeled, to account for the different deduction possibilities. In this case the 5th and 6th character of the label are used to indicate the instruction or label number within the block. BA011 is the second instruction requiring a tag in step A-01. (The first is always labeled BA01, to insure that connectors elsewhere in the program will use the right entry point.)

A—Areas.—Using the 1401, only four basic input-output areas can be defined:

* For those who mourn the loss of mnemonics, the mnemonic BACH may be used to remember this scheme.

L for Label Area
T for Tape Area
P for Punch Area
R for Read Area.

Since the read and punch areas are in fixed locations, the latter two categories are used only to indicate additional storage areas for card images other than the first.

The second character of the Area label is used to distinguish the area type. AT refers to a tape area, and AL to a label area.

The third character of the Area label is used to specify a unique character defining the file. It would ordinarily be possible to distinguish between an input and an output area for each file or to indicate the tape drive number used. Neither is practical in the 1401, since frequently an input area is also used as the output area for the same file and tape swapping precludes the use of drive number. The unique character may be assigned on the basis of priority, or on the basis of the file number on the flowchart. AR1 refers to the first card storage area used by the program, and AT2 is the second tape area.

Characters 4, 5 and 6 are used only to designate fields within the area. If it is desired to address the entire area, either for reading or writing or high speed transmission, the three digit label (AT3, AR2, etc.) is set up as its tag. The card column number or the relative location of the field may be used to designate specific fields in any area. In this case, AT1095 refers to the 95th position of the record in tape area 1, and AR2043 relates to column 43 of the card in read area 2. If the assignment of a mnemonic tag is preferred, the field name can be abbreviated into three significant characters and used, provided that a *legend of the abbreviations* is included as the first page of the block diagram.

C—Constants.—The designation for constants is the most complex but also the most helpful in debugging and program review. The second character of any C-tag always designates the type of constant:

L for Literal—an unsigned numeric constant
N for Numeric—a signed algebraic constant
C for Counter—a field into which other fields are accumulated
W for Work area—a field used as temporary storage
E for Edit word—the mask control word used in printer edit operations
T for Table entry—a storage area for a related group of data
X for Switch—an in-memory device for program switching
A for Address constant—the actual address of another variable
H for Heading—the page headings used on printed reports
M for Message—a notification to the operator of action required or taken.

The most useful and the most frequently used types are the Literal, the Counter, the Edit word and the Switch.

With the exception of the switch, address constant, message and heading, the third character designates the size of the constant. Since in the 1401 SPS System no constant can exceed 32 positions, only one character is required. The size is coded excess 9, similar to the 1401's own coding scheme for addressing the upper quadrants of memory. The presence of A-bit zoning indicates a constant ranging in size from 10 to 19, B-bit zoning indicates a tens position of 2, and AB-bit zoning indicates a 30-39 position constant except as noted:

Size Code	*Actual Size*	*Code*	*Size*	*Code*	*Size*	*Code*	*Size*
1	1	/	11	J	21	A	31
2	2	S	12	K	22	B	32
3	3	T	13	L	23	C	33
4	4	U	14	M	24	D	34
5	5	V	15	N	25	E	35
6	6	W	16	Ø	26	F	36
7	7	X	17	P	27	G	37
8	8	Y	18	Q	28	H	38
9	9	Z	19	R	29	I	39
0	10	−0	20	+0	30		

This is not as difficult as it might seem, since the 1401 uses the same code to refer to all addresses above 999. Any programmer who has to do actual testing using machine memory dumps and machine language corrections must be completely familiar with this code.

The use of the remaining characters varies somewhat with the type of constant.

CL and CN—Literal and Numeric.—Characters 4, 5, and 6 are used to show the last three significant digits of the actual constant. The numeric will have a sign over the low order position, and the literal will not:

Examples: CL11 A literal, length 1, value 1; thus the constant 1
CL3145 A literal, length 3, value 145; thus the constant 145
(Note that "CL3145–2" is equal to CL11 in the 1401, and should be used in this case)
CN24E A numeric, length 2, value 45, sign plus, the constant 45

CC and CW—Counter and Work Area.—Characters 4, 5, and 6 are used to indicate the Block Letter and Step Number of the routine in

which the information is used. In case of the counter this would mean the place where the counter was updated most frequently; the work area is used most in the routine that places information in it.

Examples: CC8B34 An 8-place counter, added to in Block B, Step 34
CWTA13 A 13-place work area used or referred to in Block A, Step 13.

The elimination of the block letter and step number from a CC or CW tag results in the labeling of a reusable constant, by definition. Thus, a CC2 is a 2 place counter which may be used by any subroutine internally, provided it need not retain any of the information after transfer of control. A loop counter would normally be reusable, since its use during the loop is controlled, and it is reset prior to loop entry.

CE—Edit Word.—In order to define, as closely as possible, the configuration of the edit word, character 4 is used to indicate the size of the data field to be edited. This also must be equal to the number of blanks and zeroes, otherwise a process check will occur. Character 5 is the number of places after the decimal, and character 6 is used as the letter of the block in which the edit word is used.

Examples: CES82 A 12-place edit word, with 8 characters of data and two places behind the decimal: $bbb,bbb.bb&
CE860 An 8 place edit word, with 6 data characters, and no decimal point: bb/bb/bb or bbb,bbb&

The use of data word size is effective when checking for accuracy of editing:

The instructions	LCA	CES82	0250
	MCE	CC9A12	0250

would immediately pinpoint error. The counter being used is defined as a 9-place counter (CC9), and the edit word only has space for 8 (CES8) data characters.

CT—Table Entry.—The third character of the table entry is the size of the entry, not the size of the entire table. The fourth character indicates the number of entries in the table and the fifth and sixth character contains the number of the entry.

Examples: CT0V01 The first entry of a table of 15 entries, with ten characters reserved for each entry
CT9A31 The last entry of a 31 entry table, with 9 characters reserved for each entry.

CX—Switch.—In order to provide proper documentation, internal switches are numbered consecutively, left justified. This makes a numeric list of switches possible and will insure that all internal switches are located in one place in memory.

Examples: CX1: Switch 1
CX15: Switch 15

CA—Address Constant.—Since the size of an address constant is always the same, the third character is used to indicate the block letter of the address. The third, fourth, fifth and sixth character refer to the Block, Step, and Instruction Number respectively.

Examples: CAB15: The address constant of the instruction at Block B, step 15.
CAD233: The address constant of the instruction at Block D, step 23, instruction 3.

CH—Heading.—Characters 3, 4 and 5 of the Heading label contain the report number of the report being prepared. Character 6 contains the heading line number if the heading has more than one line. This will enable strict control of heading print outs, insure that they are in sequence, and in multi-report producing programs will place the right heading on each report.

Examples: CHP172: The second heading line for Payroll Report No. 17. The printing instructions for this heading should always be preceded by the instructions for CHP171.
CHA12: is the only heading line for Accounting Report 12.

CM—Message.—Messages printed on the typewriter or the printer are used for control purposes. The only significant element of the message is its contents, so that a mnemonic tag may be used to indicate this. Characters 3 to 6 may be used for the mnemonic designation, or for the contents itself in case of a short message.

Examples: CMEØJ End of job message; in this case the message could be EØJ, E.Ø.J., or END ØF JØB. Since messages take valuable space in memory, and expensive typing time, the shortest message is usually preferred.

CMTPER—A Tape Error Message.—

H—Halt.—Characters 2, 3, 4, and 5 are used for the number of the halt. A halt numbering system is defined in a later section of this chapter.

The system at first appears complex, but very short experience with a standard labeling system of this kind makes it completely familiar to the user. The sudden acquisition of the ability to review easily other programs as well as ease of desk checking and testing, rapidly convinces the user of its merits. In reviewing hundreds of programs, the author has found that the total time for review is reduced almost 50% when a standard label system is used. In larger programs this factor increases to almost 60%. The writing time of programs is not increased.

Since it is possible to have exceptional conditions which do not fit the coding scheme outlined, one additional rule provides that any label which is "non-standard" must have an X in the 6th position. A special constant, of 35 positions, whose value was not significant could then be tagged as CLEXXX. Similarly, the index registers which defy labeling in any other way can be simply referred to as AINDEX BINDEX and CINDEX for numbers 1, 2, and 3 respectively.

The use of the 1401 Standard Label System outlined is illustrated by the inclusion of four sample pages of coding as Figure 4-7. The coding has been selected to show the greatest variety.

A somewhat simpler system is illustrated as Figure 4-8 below. In this case, nine alphabetic digits are available for the tag, and the first four are assigned to the program code or program number. There is no type identifier but duplication is prevented by using the characters KP to identify the Constant Pool, and using area types always distinct from possible subroutine name assignments. This is accomplished by allowing subroutine names to start only with the letters A, B, F, G, and H, and defining area types using the other letters.

Program Organization

It is necessary to indicate rules for the manner in which the coding should be organized to make up a complete program. Two kinds are required; the first sets up the original organization of the symbolic entry deck, which will then produce a listing consistent from program to program. The second relates to the set up of the object program produced by the translator. This should be done in a standard manner also so that the sequence of input information can be controlled. Both kinds of rules are illustrated below:

Symbolic Organization

1. The first page of coding of each program shall consist of comments cards which carry the following information:

Program number
Program name
Programmer name
Revision date and revision number
Console set-up conditions
Alteration switch usage
Input-output device usage
Input tapes to be mounted
Output tapes to be mounted

During testing the programmer or operator will thus be provided with the complete operating instructions as a part of the program listing. It will then act as temporary documentation in advance of the final operating manual.

2. All temporary routines used for initial housekeeping functions should be located following the identification information. This includes assignment programs, execute routines or basic housekeeping such as printer line-up, date card entry, etc.
3. The next program segments to be loaded are the main blocks of the program, in sequence by Block Letter.
4. Standard subroutines follow, in sequence by subroutine designation number.
5. All area definitions are next, in sequence by type and file identifier.
6. Constants are last, in direct label order.
7. The last item in the symbolic deck is usually a sentinel card, indicating the end of the program.

Object Organization.—Similar rules can be established for the organization of the object deck, which is in constant use by the operating department. If variable data must be added to the program or entered each day in the run, then it is best to color code segments of the deck where the variable data is entered. For example:

8. The object program deck shall be organized in the following sequence, using the color scheme noted for each section:

Program load routine	Blue
Main program—preceding operator entry	White
Main program—following operator entry	Red
Data	Manila
Last data card sentinel	Orange

LINE	COUNT	LABEL	OPERATION	(A) OPERAND ADDRESS	±	CHAR. ADJ.	IND.	(B) OPERAND ADDRESS	±	CHAR. ADJ.	IND.	d	COMMENTS
0,1,0		* EXAMPLE	OF STANDARD					LABEL					
0,2,0		*											
0,3,0		* A CLOSED DIVISION						SUBROUTINE					
0,4,0		* KNOWN	AS -BLOCK				K	- OF A FULL					PROGRAM
0,5,0		*											
0,6,0		* THIS	ROUTINE USES					A FIXED DIVISOR - CC4K07 -					
0,7,0		*											
0,8,0		* A VARIABLE DIVIDEND						ADDED TO					- CC8K -4 -
0,9,0		*											
1,0,0		* AND PRODUCES A QUOTIENT AT - CC5K -											
1,1,0		* THIS RESULT CONTAINS 4 DECIMALS THE ROUTINE											
1,2,0		* HALF ADJUSTS UPWARD THE 4TH											DECIMAL
1,3,0		*											
1,4,0		* THE ENTRIES TO THIS ROUTINE ARE -											
1,5,0		* ADD VARIABLE DIVIDEND TO CC8K - 4											
1,6,0		* BRANCH TO BK01											
1,7,0		*											
1,8,0		* AFTER EXECUTION THE QUOTIENT										X	X.XXX ADJUSTED
1,9,0		* IS MOVED FROM CC5K - 1 TO OUTPUT											
2,0,0		*											

AREA-DEFINITION CHARACTER COUNT → 1 5 10 15 20 25 30 32

630150MSP

LINE	COUNT	LABEL	OPERATION	(A) OPERAND ADDRESS	±	CHAR. ADJ.	IND.	(B) OPERAND ADDRESS	±	CHAR. ADJ.	IND.	d	COMMENTS
0,1,0		* BLOCK	K K - DIVIDE				S	SUBROUTINE					
0,2,0		*											
0,3,0		BK01	SBR	BK21	+	3							SAVE EXIT
0,4,0			MZ	CC8K	-	4		CC1K					SAVE DVDEND SIGN
0,5,0			MZ	CL11				CC8K	-	4			DEZONE DVDEND
0,6,0			S	CINDEX	+	1							SET SHIFT X TO 0
0,7,0			S	CC5K									SET QUOT TO ZERO
0,8,0		* K- TEST	THAT DVDEND					IS NOW -ZERO					& DVSOR GREATR
0,9,0			MCS	CC8K	-	4		CC4K05					IF ZERO DVDEND
1,0,0			B	BK18				CC4K05					ZERO QUOT AT K18
1,1,0			C	CC8K	-	4		CC4K07					C DVDEND/DVSOR
1,2,0			B	BK13								T	LO DVSOR ERR K13
1,3,0		* K- DIVIDEND 0K -					P	ROCESS		INIT		I	AL SHIFTS
1,4,0		BK08	C	CC8K	-	43		CC4K07					C DVDEND/DVSOR
1,5,0			B	BK14								U	IF HI SHIFT -K14
1,6,0		* K INITIAL SHIFTING						& OVER, ADJUST					SHIFT INDEX
1,7,0			S	CL210				CINDEX	+	1			EXCESS SHFT ADJ
1,8,0		* K- TRAIL SUBTRACT						& TEST	DIV	DEND			FOR MINUS
1,9,0		BK10	S	CC4K07				CC8K	-	43			TRIAL -DIVSOR, IF
2,0,0			BWZ	BK15				CC8K	-	43	K		DVDEND CR TO K15
210			A	CL11				CC5K	-	43			-NO ADD TO QUOT
220			B	BK10									& REPEAT SUB-K10

AREA-DEFINITION CHARACTER COUNT → 1 5 10 15 20 25 30 32

630150MSP

Fig. 4-7. Example of IBM 1401 Standard Label System.

LINE	COUNT	LABEL	OPERATION	(A) OPERAND ADDRESS	±	CHAR. ADJ.	IND.	(B) OPERAND ADDRESS	±	CHAR. ADJ.	IND.	d	COMMENTS
3 5	6 7	8 13	14 16	17	23		27	28	34		38	39	40 55
010		*K-HALT		#123		FØR	E	XCESS		DIVID	E		ND ERRØR
020		BK13	H	0123				0123					H123 DIVIDE ERR
030			B	BK18									ØUT TØ EXIT -K18
040		*K-AN	I	NITIAL		SHIF	T	IS		REQUIRE			D
050		BK14	A	CL11				CINDEX					SHIFT & RETURN
060			B	BK08									TØ C/TEST AT K08
070		*K-C	ØRRECT	FØR		ØVE	R	DRAW	&	TEST			FØR DIV END
080		BK15	A	CC4K07				CC8K	-	43			CORRECTIØN IF
090			BWZ	BK18				CC8K	-	43			STILL CR END-K18
100			A	CL11				CINDEX					SHIFT IF SHIFT 5
110			B	BK18				CINDEX				5	DIV ØVER-TØ -K18
120			B	BK10									-NØ TØ TRIAL-K10
130		*K-DIVI	SION	ENDED		-1/2	ADJ	QUØT					-AND COMPLETE
140		BK18	9	CC8K									ZERØ DVDEND AREA
150			A	CL15				CC5K					HALF ADJUST QUØT
160			MZ	CC1K				CC5K	-	1			SIGN RESULT
170		BK21	B	0000									VARIABLE EXIT
180													
190													
200													

AREA-DEFINITION CHARACTER COUNT → 1 5 10 15 20 25 30 32

630150MSP

LINE	COUNT	LABEL	OPERATION	(A) OPERAND ADDRESS	±	CHAR. ADJ.	IND.	(B) OPERAND ADDRESS	±	CHAR. ADJ.	IND.	d	COMMENTS
3 5	6 7	8 13	14 16	17	23		27	28	34		38	39	40 55
010		*BLØCK	K	-COUNTERS									
020		*											
030	08	CC8K	DCW	*	+	0000	0	0000					DIVDEND WØRKAREA
040	01	CC1K	DCW	*									SAVE SIGN CTR
050	04	CC4K05	DCW	*									DVDEND ZERØ TEST
060	04	CC4K07	DCW	*	+	0000							SET WITH +DIVSØR
070	05	CC5K	DCW	*	+	0000	0						QUØTIENT CTR
080		*											
090		*K-CØNSTANT	LITERALS										
100		*											
110	01	CL11	DCW	*		1							
120	01	CL15	DCW	*		5							
130	02	CL210	DCW	*		10							
140		*											
150		*K-INDEX	WØRD	-USUALLY		CØDED							IN BEGINNING
160	03	CINDEX	DCW	0099		000							
170		*											
180		*NØTE	THAT	THIS	SUB	RØUTINE		DESTRØYS		THE	PRIØR		
190		*CØNTENTS	ØF	INDEX 3									
200		*											

AREA-DEFINITION CHARACTER COUNT → 1 5 10 15 20 25 30 32

630150MSP

Fig. 4-7. (cont.)

CHARACTERS	9 8 7	6	5 4	3 2	1
INSTRUCTION	PROGRAM CODE		SUB-ROUTINE NAME	STEP NUMBER	IN-STRUC-TION LETTER
CONSTANT	PROGRAM CODE		K P	CONSTANT NUMBER	
AREA	PROGRAM CODE		AREA TYPE	AREA NUMBER	FIELD CODE

Fig. 4-8. Standard Label System. (*Courtesy*, The Bowery Savings Bank)

Figure 4-9 shows a method of graphically illustrating the composition of an object deck. Figure 4-10 shows a similar method for an object tape.

PROGRAM ORGANIZATION AND DOCUMENTATION

OBJECT DECK ORGANIZATION

ALL COMPLETED OPERATING PROGRAMS WILL BE STORED IN MULTIPLE LOAD CARDS. THE OPERATING DECK AS OBJECT DECK WILL BE ORGANIZED AS FOLLOWS FOR STANDARD OPERATING PRACTICES.

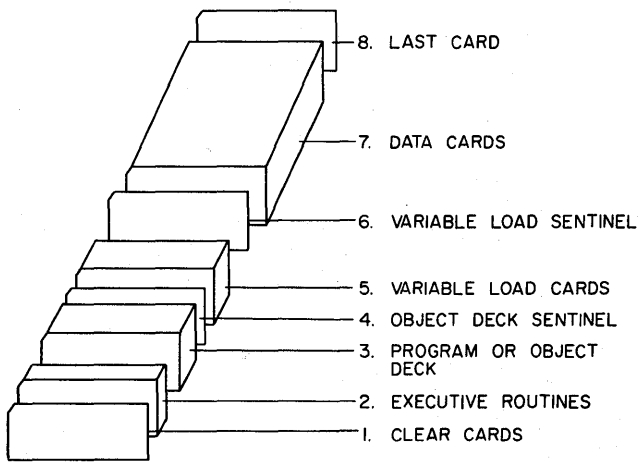


Fig. 4-9. Program Format and Organization. (*Courtesy*, Bulova Watch Company)

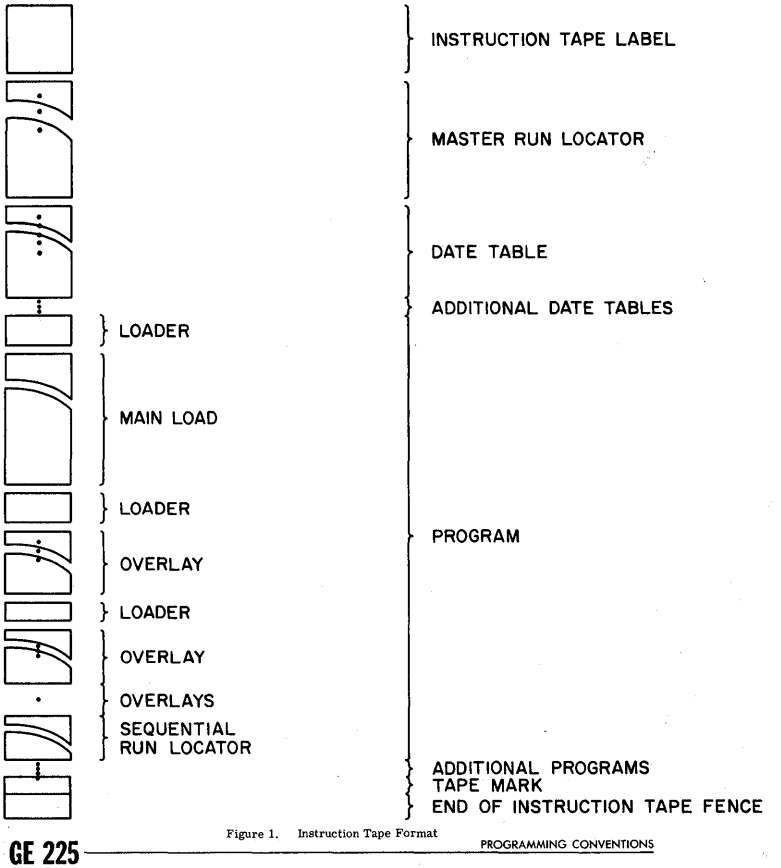


Fig. 4-10. Instruction Tape Format. (*Courtesy*, Computer Department, General Electric Company)

PROGRAMMING RULES

Rules should be established for the actual coding in order to prevent common errors and to improve quality. These can be grouped in the following categories:

- General programming rules
- Machine-imposed rules
- Symbolic translator rules

General Programming Rules

Typical general programming rules are indicated below.

1. Never use a part of an instruction as a constant. This creates confusion and may cause errors if program changes are required.
2. In establishing a loop, *always* reset the loop at the beginning, not at the end.
3. Never assume that memory has been cleared (set to blanks or zeroes) at the beginning of the program. Always clear the necessary areas during initialization.
4. Always set counters and switches to their initial conditions during the housekeeping operation.
5. When reading input created externally to the computer, always check the validity of the information.
    a. Make sure that all fields which should be punched are punched
    b. Check all numeric fields for double punching or blanks
    c. Check the input sequence
    d. Check the consistency of fields by testing for all possible entries, or by checking for a range.
6. *Never* use address adjustment to refer to instructions. Always use a label.

A number of general programming rules which relate to the operation of the machine. For example:

7. Always rewind all tapes before their initial use; in order to allow maximum set-up time, however, rewind only those which are immediately required. Do not attempt to use a tape that was not rewound.
8. Always restore the printer to its top line using programming. Do not rely on an operator to save one instruction in housekeeping.
9. Always validate the header labels on tape; check input for the correct file identification and the correct reel sequence. The output must also be checked for the expiration of the usable date.
10. Always use a separate printer line-up routine when using imprinted forms; print the form number and date as part of the initial line-up line.
11. Always print a heading on blank paper; this is an excellent way to sell management on the effective use of the system.

Machine-Imposed Rules

Rules imposed by the design of the machine should be divided into two categories for emphasis. In the first category are the machine restrictions that must be observed; the second category should be presented as a list of common causes of errors. As examples:

For the IBM 1401:

1. When using Edit (E), the number of blanks and zeroes in the control field may not be less than the number of characters in the data field.
2. Using "Advanced Programming," the Branch (B) instruction places the I register contents in the B register. The Print and Branch (2), Read and Branch (1) and all other combined execution and branch instructions do *not* place the I register contents in B.
3. An unequal Compare (C) will result when two fields of different lengths are compared, and the B-field is the longer field.
4. Arithmetic operations will remove all zoning from the B-field, except for overflow zoning in the high-order position and sign control in the low-order position.
5. In a single digit accumulator the zoning acts as sign control, NOT as overflow. The sign will therefore remain the same, unless a complement operation occurs. Therefore: $9 + 1 = 0$, not a record mark.

For the IBM 1401, some common causes of errors:

1. Editing a given size data field into a shorter edit control word
2. Failure to have an ending wordmark after a combined branch instruction
3. Address modification using Add, without setting wordmarks in the instruction
4. Incorrect address adjustment
5. MCM transmit without a record mark between the starting and ending address
6. Decrementation of a register below 0000 or above the upper memory limit through incorrect addressing
7. Using an operation code of blank or 0, through incorrect branching.

For the IBM 705, some interesting statistics in testing:

Causes of errors in initial tests:

Sign Check	0905	42%
Overflow	0904	16%
Invalid Op Code or Address	0900	12%
Runaway Transmit		9%
I/O Not Available		6%
Invalid Operation as reading the typewriter writing on the card reader writing on an input tape		5%
Other		10%

Symbolic Translator Rules

In order to properly use the symbolic translator provided with most machines, a fairly rigid set of rules is established. Many of these rules

are included as a part of the documentation of the assembly program; others are derived through usage. It is good practice to summarize these as a part of the rules section of the Standards Manual.

As an example, the following represents a small part of the rules used for the Univac 490 Symbolic Program (SPURT):

1. The operator must always be the first word of an L_0 operation statement.
2. The *j*-designator operand, if used, appears as the last operand of a mono-operation.
3. The operator and each of its operands are separated with a point separator, •
4. When used, the ENTRY operation must have a label.
5. The V_0 operand in mono-operations is either a register designation or a modifying expression.
6. A *j*-designator is required in all CØM operations, including CØM-MASK.
7. A *j*-designator is not allowed if the V_0 operand is B_1 through B_7 or C0 to C15 in the operations: ENT, STR, BJP, BSK, CL, IN, ØUT, EX*FCT.
8. All operations of the Univac 490 referencing channels must have OCTAL channel designators: C0-C7, C10-C17.
9. The contents of register Q cannot be transferred directly to a B-register. It may only be transferred indirectly using a temporary storage location.
10. Do not use a conditional SKIP instruction immediately before a poly-operation or a macro-instruction, unless it will generate only *one* machine instruction.

Many other rules are possible; the most important factor to be considered is that the rules are documented in a central location so that novice programmers can benefit from the experience of the rest of the staff.

AUDIT AND CONTROL STANDARDS

One of the most important attributes of effective computer usage is the maintenance of accurate control statistics on the data being processed. The method used will vary from installation to installation, and from application to application. A bank has certain legal regulations with which it must comply but it does not need to retain these same audit controls when it processes its own payroll, and a completely different set of controls are required for non-critical operations. It is therefore difficult to establish a general set of standards for auditing. However, some specific rules that could be applied to most installations are:

1. All operator actions which can have any effect on the information being processed must be recorded on a typewriter or magnetic tape log.
2. There must always be two operators, or one operator and one supervisor, in the machine room when critical data is being processed, to avoid misappropriation.

3. All information entered into the system which alters any type of money balance must be recorded separately on an audit control tape or ledger record.
4. All journals, ledgers, or other critical listings must have a sequential numeric page number. Page 001 will be the first page, and the last data page shall force a separate page to be created carrying the message: "LAST PAGE *XXX*."
5. All data files must have a minimum of two controls, verified at predetermined control "breaks." One of these controls shall be a sum total of all the money being processed; the other shall be a hash total of the field which controls the sequence of the file.
6. In all financial programs the audit trail shall be clearly defined and noted as a separate part of the documentation.
7. Machine checkpoints shall be taken at all control points.
8. All magnetic tape files shall be retained a minimum of two cycles for purposes of file protection and back-up.

CONTROL OVER OPERATIONS

The programmer, in creating an "air-tight" logical framework, is directly responsible for the prevention of operator errors. It is foolish to allow an operator to make a major decision time and again, if the programmer can make the same decision once and ask the machine to execute it. The old-fashioned method of stopping the machine to let the operator make a correction can no longer be used. Machine time is more expensive, but more important the effect of one error can be staggering in a complex system which has been designed around the assumption of machine (but not operator) infallibility. For this reason, programming management must very carefully consider the standards it establishes for all conditions where operator action is required, including:

- Set-Up and Take-Down
- Programmed Halts
- Machine Failure Halts
- Messages
- Decisions

Set-Up and Take-Down

The instructions given to the operator to set up and take down the machine are discussed in more detail in Chapter V—Documentation Standards, page 126. To verify that the set-up has been properly accomplished, the following should be included in the program:

Printer: Restore to the top line
Print a "line-up" line and include the form number

Tape: Check labels on input and output after rewinding
Check dial settings, if possible, for availability and possible duplicates

Cards: If punching, create an identification card as the first card of the file, to insure that there are cards in the punch, and to signal file disposition
If reading, check all cards and demand the use of sentinel cards wherever possible

Console: Always type the program name and date
Verify switch settings by typing the actions which will result, and allowing changes:
CONSOLE SWITCH 1 ON — SUSPEND OUTPUT REPORT
CONSOLE SWITCH 2 OFF—ALTERNATE MASTER INPUTS
STOP NO. 1111
NO CHANGES

Paper Tape: Check for header identification

The same type of actions can be taken to force a correct take-down procedure:

Tape: Rewind and unload all finished files
Type a message to signal "writeinhibit" ring removal

Card punch: Create a final blank card to insure that the last data card is not left in the punch

Printer: Restore the form several times, to make sure the paper will not be torn, or left in the carriage.

Console: Type END OF JOB PROGRAM *XXXX*
STOP NO. 9999

Programmed Halts

Most systems have an operation code which stops the machine. The stop may be occasioned by an error condition, the requirement for an external decision, or the termination of processing, such as an end-of-phase or end-of-job. It is the responsibility of the programmer to clearly identify the stop condition, so that the operator can determine its cause and the action required. Programmed halts are therefore given an identifying number, which can be related to the documentation.

The method of numbering the halt will vary depending on the machine display available. If the system is equipped with a typewriter or monitor printer, the halt should be identified by typing a message containing the halt number and the cause prior to stopping. This will provide a log, as well as a means of identification.

If a typewriter does not exist the most common way to number halts is to use the contents of one of the displayed registers as the halt number.

The location register could be used, except that a program reassembly will almost always change the location of the halt. This will affect the documentation and the operator recognition which has been established for each program. In any case, a register should be used whose contents are immediately available to the operator, *without* any further console manipulation. If it then becomes necessary to use the location register, as in the 1401, it is quite simple to locate all of the halts at the front of the program, so that reassembly will have no effect.

Standard Halts

It is a very common practice for programmers to number halts in a fairly random manner. Some use a consecutive system, others use simple configurations of numbers, and others use their own telephone number to signal critical conditions. Even though this practice has certain direct advantages it is better to use a system of standard halts which will enable rapid recognition by the operator.

Halts should be standardized in two categories. Each installation should establish a series of halts which are standard for *all* programs and are included as a standard package in all programs. It is obvious that all programs will have an "end-of-job" halt; it should not be discretionary with the programmer to assign a unique number since it should be rapidly recognized. Other standard halts include:

- Sequence error
- Printer line-up
- Tape reading error
- Tape writing error
- Input tape end
- Output tape end
- Tape label error, input
- Tape label error, output
- End of phase

The second category of halts pertains to those not included in all programs. These halts are unique to the program, but certain numbering rules can provide identification in relation to the type of halt, its location, type of action, etc. For example, if a four digit halt number were used:

Digit 1 could indicate the action required
1 Normal termination
2 Interrupted termination
3 Failure to complete—terminate for rerun

4 Approval required
5 Decision required

Digit 2 could indicate the program section (and the documentation location)

1 Standard Halt
2 Initialization
3 Input-Output Subroutine
4 Other Subroutine
5 Main Program

Digit 3 could indicate the type of halt—

1 End of a routine, subroutine, phase or program
2 Error—Data
3 Error—Machine
4 Input
5 Output

The last digit can be used to further break down the cause of the halt, or the action required. A more common practice is to indicate the address of the unit involved in the halt, if any. For a six-tape system, the last digit might be used for:

0 No peripheral equipment involved
1-6 Tape units 1 through 6
7 Paper tape reader
8 Printer
9 Card reader

Machine Failure Halts

Non-programmed halts or halts caused by machine failure, are outside of the jurisdiction of the programmer. If a halt of this kind occurs, it should be standard practice for the operator to call his supervisor. If correction or continuation is possible the run should be continued. If not, the engineer should be notified and the run terminated. Other operating procedures are discussed in Chapter VI.

Messages

Typewriter or printer messages are also used to notify the operator of the progress of the program and its problems. Messages are further used to create a log of all occurrences, starting with the program name and the date, and ending with the end of job type out.

Such messages provide an extremely valuable medium of man-machine communication, because of their clarity and their permanence. Messages are expensive however because they absorb a great deal of memory space and take a long time to be typed out. As a result, the following kind of rules should be applied:

1. Messages that may be typed out must be clear but concise. Avoid unnecessary connectives and adjectives. Abbreviate but do not lose meaning. Always indicate pertinent information such as tape unit number for both operational and engineering purposes.
2. The following abbreviations used in messages will have the meaning indicated:

H	Halt	ØK	Approve
EØF	End of File	IND	Indicator
EØR	End of Reel	ALT	Alteration Switch
TP	Tape	DT	Date
CD	Card	STD	Standard
PH	Phase	IP	Input
RCD	Record	ØP	Output
ERR	Error		

3. The following messages will be standard:

```
PROGRAM XXXX
MM/DD/YY
OK PRINTER
LABEL ERR: TP N, FILE NO: YYYY
TP ERR: N
H NNNN-CAUSE
REMOVE REEL N, TP N
RCD COUNT: NNNNNNNN
END PH. N
* END OF JOB * PROGRAM XXXX
```

4. All error condition messages will be indented one space.
5. All messages that require operator action will be indented two spaces.

Decisions

As a general rule for programmers, operator decisions should be kept to a minimum. Other rules to be kept in mind include:

1. Decisions required from the operator should be clearly stated. Decisions should always be reduced to simple yes/no possibilities.
2. If decisions are to be typed in by the operator, always try to cover all possibilities. Do not assume that if the answer is not YES, it must be NO. It could have been YEP, or YEA or any of a number of other spellings, indicating operator error.

3. If the decision is to be communicated through the setting of console switches, always DEMAND a positive action. If the choice were to turn ON a switch and depress the start key, or to leave the switch OFF and depress the start key, the operator might inadvertently depress the start key without having made a decision. In this case two switches should be used, one for YES and one for NO. If the decision is entered by console key always type or record the result of the key setting for logging purposes.

4. In entering variable data such as a date or a starting check number always use a hard copy medium rather than the console. Since the validity of the information cannot be checked, as in the case of YES and NO, a punched card or paper tape should be used to avoid transposition.

5. In entering decisions through the typewriter always require more than one character for the answer. One character can be easily generated by someone's elbow on the keyboard.

If operator decisions can be avoided altogether the amount of rerun time will be reduced.

CHAPTER SUMMARY

Programming standards should be established for each element of the programming task. Included in these standards are

- Usage of the Job Specification Manual
- Logical analysis: Block diagramming method and complexity
- Character writing conventions and symbology
- Standards for coding: the use of mnemonics (not recommended)
  the coding format
  the coding method
  standard labels (recommended)
  program organization
- Programming rules: for day-to-day programming
  machine violations
  symbolic rules
- Audit and control
- Control over operations, halts, messages, and operator actions

The basic premise of the Chapter is that it is mandatory to establish and document a series of rules for all aspects of programming, without exception. These programming rules are continued in Chapter V.

Questions for Review

1. Indicate the factors required to establish methods standards for programming.
2. Design a standard label system for a machine with which you are familiar.
3. Carry a small program from block diagram through programming, using the standard rules and standard labels developed in this chapter.
4. What audit standards would you recommend for your management?
5. Indicate the type of rules which are necessary for a training manual for programmers and for operators.
6. Draw a symbolic representation of the organization of your symbolic entry cards.
7. Develop a list of standard abbreviations.
8. Indicate the most important rules in programming.
9. Develop your own halt numbering system.

Chapter V

METHODS STANDARDS: PROGRAMMING, PART 2

INTRODUCTION

Chapter IV discussed the standards that should be established for logical analysis and coding. This chapter continues with programming methods standards for the functions of program testing and program documentation. Programming is often equated with coding, but the more important aspects of programming are logical analysis, testing, and ultimately documentation. Standards in these areas are therefore necessary and require considerable thought on the part of the user.

TESTING AND PROGRAM VALIDATION

To err is human, and to err in a program is normal. Most programmers writing their first program assume that it will work the first time placed on the machine. The fact that it does not, and that it may not for the next fifty times, requires rigid standards in testing. Program testing has two basic purposes:

- To determine that the program has been coded correctly and that the coding matches the logical design
- To determine that the logical design matches the basic requirements of the job, as set down in the job specification

Because a program can contain as many as 30,000 distinct and separate instructions, and because a program must handle all possible input and output conditions, the number of errors may be quite large. Program errors fall in the following categories:

- Errors in logic
- Clerical errors
- Misinterpretation of the machine's functions
- Misinterpretation of the requirements of the job
- Systems analysis errors

The number of errors in a program will average one for each 125 instructions, assuming that the programmer has been reasonably careful in his coding system. The number of permutations and combinations of conditions in a program may reach into the billions before each possibility has been thoroughly checked out. It is therefore a practical impossibility to check out each and every possible combination of conditions—the effort would take years, even in the simplest program. As a result, it is quite possible for errors to remain latent for a number of years, suddenly appearing when a particular combination is reached which had not previously occurred. In one such instance, a large installation had been running a weekly payroll program for over three years, handling approximately 30,000 employees. After three years of operation, the program suddenly failed. Analysis disclosed that the failure occurred because two employees whose names were contiguous on the payroll master file had been married during the same payroll period. This had never occurred before and the program had never been tested for this somewhat remote possibility. A more famous error occurred in the Ballistic Missile Early Warning System at Thule, Greenland, when a computer program which had been in operation only a few days interpreted the reflection of the moon as a missile attack on the United States.

More recently, an $18 million missile was destroyed because the guidance program was missing a significant *x* punch in a statement; this directed the missile to zig-zag in error. This occurred despite the fact that hundreds of experts had carefully checked the statements in the program, time and again.

The average installation will not be faced with problems of similar magnitude. None the less, latent program errors will remain in operating programs, and their occurrence should be minimized by complete and thorough testing. The fact that the program is operative and reaches end-of-job satisfactorily does not mean that all of the exception conditions, and their permutations and combinations, have been tested. Quite the contrary, many programs reach end-of-job after very few tests, since the "straight-line" part of the program is often simplest. However, the exceptions programmed to deal with a minimal percentage of the input, account for a large percentage of the instructions. It is therefore quite possible to reach the end-of-job halt with only 10% of the program checked out.

TYPES OF TESTING

Many testing procedures must be performed to find the greatest number of errors. These procedures occur in a natural sequence after completion of the coding:

Desk Checking

The first step in the complete checkout procedure consists of a detailed review of the program by the programmer and/or the project coordinator or the supervisor. Desk checking itself consists of two parts; the first is a review of the general logic for completeness; i.e., that there are no logical loopholes or flaws in the design. At this time the program is also reviewed for quality, making sure that good techniques have been used, and that the machine's capabilities have not been misinterpreted. The second part of the desk check is the "dry run," in which some sample data is used, and the programmer "plays computer." He checks the manner in which the program handles the data, keeping track on paper of the status of pertinent registers, triggers, and indicators.

Program Preparation

After the coding has been reviewed and desk checked, the program is ready for input preparation. Some computers allow the symbolic program to be entered directly from paper or magnetic tape. The most common medium is cards, and most programs must be keypunched.

If the testing is to be done on a machine borrowed from the manufacturer, machine time will be scarce or expensive. In this case it often pays to perform some punched card machine operations on the program deck to catch clerical and other obvious errors. It is possible, for example, to sort the deck by operation code and to match the deck on a collator against a master deck of acceptable codes; this will insure that the operation codes used are all valid and will be properly translated by the assembly program. This type of procedure often saves computer time at the much lower expense of punched card operator time.

Assembly or Compilation

The third step in the testing procedure is to assemble the symbolic entries and to translate them into machine language. The assembly program, if well-designed, has the power to check or validate single instructions and sometimes combinations. All assembly programs will

validate the operation codes and operands, since an invalid one would prevent translation; many also incorporate the ability to catch machine violations, or to flag instructions that *might* cause a problem.

A further extension of this method of machine validation is the use of an analyzer. An analyzer is a program that scans the instructions in an object program and determines if they are valid or could lead to problems. A static analyzer operates on the program by analyzing each instruction in turn; a dynamic analyzer will actually operate the program, but prior to executing each instruction will check the validity of the operation. If the operation is considered valid, it is executed; if it is found to be invalid or may cause a destruction of other instructions, it is flagged on a listing and bypassed. Analyzers are sometimes provided by the manufacturer, but more often it is up to the user to develop his own techniques.

Program Testing

After the assembly errors have been corrected, which may have required a second assembly, the program is ready for the next testing phase. This is program testing, in which the program is placed in the machine and allowed to operate on fabricated test data, designed to simulate a number of conditions of input. The number of test "shots" required will vary. Smaller programs are completed in from 3 to 10 shots but a larger program may require 100 distinct test shots to purge the program of all possible errors. Much of this depends on the thoroughness of the programmer in coding and in desk checking. A sloppy or "fast" programmer loses a great deal of time in testing but a "tricky" programmer is sometimes endlessly trapped in his own schemes.

Errors found in program testing appear in many different ways. For example:

- An endless loop may be caused by a logical flaw or by an incorrect test for the end of loop signal or count
- The machine may attempt to reference areas beyond its memory; this may be caused by faulty address modification
- The machine may stop in an illegal operation code; perhaps caused by a wrong jump or by the partial destruction of memory
- The machine may stop because of illegal conditions caused by clerical errors, misassumptions, or execution of invalid instructions

Regardless of the way in which the error appears, it must be carefully traced to its cause, which may be far removed from the point at which it appears. This is most often begun with a print-out of memory (a

"dump") at the point of error occurrence. The error can then be traced, after the machine test, and carefully and accurately corrected. Beginning programmers often create added errors when making their correction. Their inexperience with "in-memory" corrections ("patches"), sometimes causes errors in the use of machine addresses or locations.

Production or Volume Testing

After the program can operate correctly on all of the fabricated test data, it has essentially been determined that the coding matches the logical analysis, i.e., that the program works in the manner in which the programmer has decided to make it work. In order to further validate the fact that the design matches the requirements, the program is production-tested using "live" or actual operating data. This will insure that the assumptions of layout are correct, and that provision has been made for all conditions of input. The systems analyst should assist in the production testing, so that he can test his own assumptions about the way in which the program should operate.

Systems Testing

After completion of production testing, it can be assumed that the program is in operating condition. In those cases where the program is only a small part of the overall system, as in most commercial and some engineering applications, the entire system must be subjected to a rigid test. This test will determine if the linkage between programs has been correctly made; i.e., where the ouput of one program becomes the input of the next program, the assumptions made in both programs must be valid—both programs must use the same layout. It is also necessary to determine that all conditions of invalid input are checked for in the system. If the system requires a file updating, a separate input conversion program can validate the initial punching of the cards. This conversion routine cannot check on the validity of the assigned account number; only the updating program can do this, by proper reference to the master input file. The systems analyst should also take part in the systems test.

Parallel Operation

The final test of the entire system of programs occurs after conversion, when the old system is run in parallel with the new computer system. This means that both systems produce output on a daily or other regular basis. Initially the output of the old system is assumed to be correct and is used and the new output is merely validated. After the

new output has been accurate for a period of time, it is then substituted as the "prime" output and the old is used as a check. The old system is finally terminated and full computer operation begins.

Engineering applications usually do not have a parallel operation. Part of the overall check-out system is to develop a sufficient number of cases in order to manually determine that the formulas have been correctly translated. If the initial formulas are not correct or the mathematical assumptions are invalid, it becomes the responsibility of the engineer to validate the program output.

TESTING STANDARDS

Methods standards established for all phases of testing must take into account the complexities of the machine and the system involved. General rules can be established, however, which are applicable to the entire testing program. A number of suggested rules are given below.

Desk Checking

1. After completion of coding the programmer shall completely desk check his program. This shall be done in the following sequence:
   a. The program shall be checked for clerical errors, missing labels and general legibility of symbols.
   b. Taking a minimum of two sample cases, the programmer shall trace the flow of the program using form No. *XXXX* [See Figure 5-1] to keep track of the conditions of the registers at each step.

LOCATION	CONTENTS REGISTER A	CONTENTS REGISTER B	INDEX 1	INDEX 2	MEMORY LOCATION TO	MEMORY LOCATION TO	SWITCHES SET

Fig. 5-1. Dry Run Tracing.

2. The program shall be reviewed by a second person designated by the programming manager. This person may be a programming supervisor, the project coordinator, or another programmer. The purposes of the review are to:
   Familiarize a second person with the program
   Minimize the number of errors
   Optimize the techniques used
   Educate both programmers to each other's techniques and methods
   Review the adequacy of the logic

Program Preparation

3. After the program has been punched, it shall be completely key-verified and interpreted.

4. Using the standard labeling system [as outlined in Chapter IV], perform EAM operations on the symbolic entry cards, in order to provide [for a 1401 program]:

Page and line listing
Listing by A-operand
Listing by B-operand
List of labels
List of halts
List of valid operation codes
List of d-modifiers
Validity of indexing
Validity of counts and record lengths against the standard tags.

Any standard labeling system lends itself well to EAM validation of input items, since the tagged items can be readily sorted by type, and checked against the requirements established for this type. The listings, created in order by the various operands and other characteristics, will be of great value in the ultimate machine testing, when it is necessary to determine all of the references to a tagged item. There are some computers whose assembly programs will provide complete cross-reference listings, in which case this phase may be skipped.

5. All programs shall be assembled using the standard symbolic program. All errors flagged by this program shall be completely investigated and corrected, if necessary. If more than 10 errors are found by the assembly, the program shall be reassembled before it may be tested.

a. Reassemblies shall be made periodically, based on the number of errors found. A reassembly shall be made after every 10 test shots, or after the number of machine language correction card entries exceeds 25, whichever occurs first. After the program is completely systems tested, a final assembly shall be made. The output of the final assembly shall be retested using the complete set of test data.

Test Data Preparation

6. Test data to be prepared falls into two categories:
   a. Data necessary to check out each block of the program
   b. Data necessary to validate the manner in which the system requirement is met.

7. The programmer is responsible for creating the test data referenced in 6*a*, the systems analyst must prepare the data in 6*b*.

8. In order to create the data referred to in 6*a*, the programmer shall use the micro-block diagram. To assure that each condition is properly tested, for each block the programmer shall prepare one copy of the Test Case Preparation form [See Figure 5-2] as follows:

a. Fill in the heading information

b. For each conditional test, with more than one choice, the programmer shall use two lines of the form. Test cases shall be numbered consecutively, and the conditions tested shall be indicated. If the decision symbol states: "Compare net fee to $1000," the programmer shall prepare a case with a net fee of under $1,000 and a second case with a net fee of over $1,000. The block diagram shall be used to insure that a test case exists for each path which the program may be forced to take.

9. The data referred to in 6*b* shall be prepared by the systems analyst using "live" or actual data obtained from the existing system, reformatted as necessary.

L. TEST CASE PREPARATION

Program Name _____ Date _____

Program No. _____ Programmer _____ Format Item _____

Type of Data _____

Test Case No.	Conditions	Prog. Ref.		Errata Addr.	Word 1	Word 2	Word 3	Word 4	Word 5	Word 6	Word 7
			A								
			A								
			A								
			A								
			A								
			A								
			A								
			A								
			A								
			A								
			A								
			A								
			A								
			A								
	Card Cols. ____				6-10	11-20	21-30	31-40	41-50	51-60	61-70	71-80

Fig. 5-2. Test Case Preparation. (*Courtesy*, The Bowery Savings Bank)

Test Data Sequencing

10. In the initial tests the test data shall be organized in sequence by block; this means that the test data will be set up in such a way that an error is pinpointed to the block in which it occurs, based on the test data item being processed. If a tape updating is being tested, for example, all of the "new accounts" should be tested first, followed by all of the "delete

accounts," followed by the "changed accounts." If an error occurs during processing, the block in which it has occurred is pinpointed by the transaction code which is being processed.

11. After the initial tests have proven the validity of the individual program blocks, the test data shall be reorganized, to check out the linkages between the various blocks.

Program Testing

12. All program testing shall be performed under control of a monitor system. It shall be the function of the monitor system to:

Distribute test data onto the necessary input tapes from tape
Generate repetitive test data from a format sample, where required
Load the object program
After hang-up, provide automatic linkage to a memory print program, followed by a tape print program; to print all output on one tape for ultimate off-line printing
Prevent unauthorized "console debugging."

13. Prior to all tests on the machine, the programmer shall fill in a Test Plan form [Figure 5-3] indicating the anticipated stops and the actions required when the stops are reached. The form has been designed to facilitate testing whether or not the programmer is at the console. If a memory dump is desired at the anticipated stop point, it is so indicated and the test operator uses the monitor linkage to create a memory dump at that point. At its completion, the monitor restores memory and returns to the point at which the stop was reached to resume testing.

TEST PLAN

PROGRAM _____ DATE _____

PROGRAMMER _____ TIME _____ AM PM CHECK ONE

SEQ STOP NUMBER	STOP TYPE: ADDR HALT AT	STOP TYPE: PROG HALT #	STOP TYPE: OTHER	POINT IN THE PROGRAM	MEMORY DUMP YES / NO	NEXT ACTION PLANNED	STOP REACHED CHECK

Fig. 5-3. Test Plan. (*Courtesy*, The Diebold Group, Inc.)

14. During testing the test operator, or the programmer, will fill out a Test Results Summary form. [See Figure 5-4.] This form indicates the stops which occurred and the sequence in which they occurred. The conditions of

the console [See Figure 5-4, designed for the 1401 console] are indicated at each stop point, and the actions taken are faithfully recorded, so that the test conditions can be reconstructed and the error rapidly found.

15. The second half of the Test Results Summary is filled out by the programmer after he has corrected his errors. For each stop which has occurred, the programmer indicates the cause of the condition, and the action which he has taken to prevent its reoccurrence, if it is an error. The main purpose of this form is to prevent programmers from ignoring error conditions, in the faint hope that they will not occur a second time. The form also serves to indicate error causes, which will enable supervisory analysis after test completion, to determine

The number of test shots which were used
The number of errors which occurred, by type
- coding errors
- clerical errors
- logic errors
- patching errors
- operator errors
- set-up or data errors.

The total machine time used for testing.

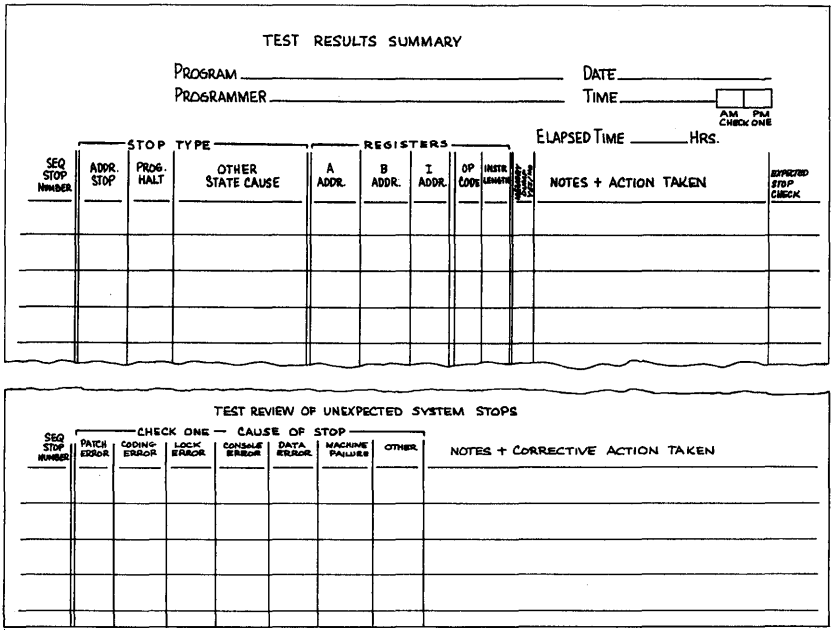


Fig. 5-4. Test Results Summary. (*Courtesy*, The Diebold Group, Inc.)

16. All corrections made during testing will be made in machine language, unless the number of instructions required for a single correction exceeds 50.

17. At the same time that corrections to the program are made in machine language, the corrections which correct the symbolic entry deck also shall be

made. The corrections shall be made on a dual purpose form [See Figure 5-5] which insures that for each memory correction the corresponding symbolic entry is also made. The corrections shall also be noted on the most recent assembly listing.

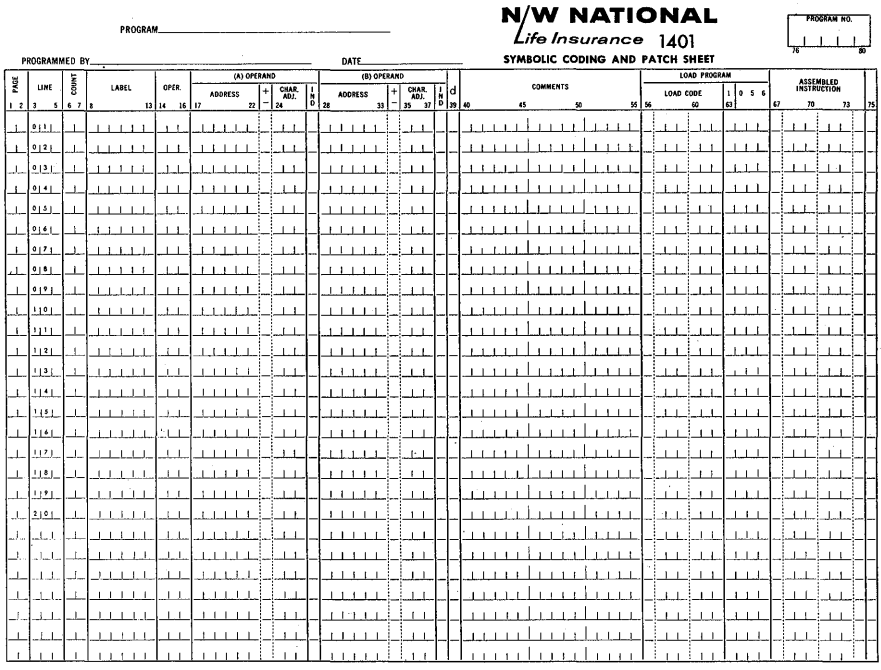


Fig. 5-5. Symbolic Coding and Patch Sheet. (*Courtesy*, Northwestern National Life Insurance Company)

18. The program shall be reassembled each time that the total number of individual patch cards exceeds 100.

19. Retention of test materials shall be limited to:
    - *a.* The Test Plan for each test sequenced by date and time
    - *b.* The Test Results Summary, in the same order
    - *c.* The last two memory prints at hang-up time
    - *d.* The last memory print taken at the end of program loading
    - *e.* One copy of the input data listing(s)
    - *f.* One copy of the most recent output listing(s).

20. To facilitate in-memory corrections, when organizing a program prior to assembly, a separate area, preferably in lower memory, will be set aside for corrections. This area shall not exceed the area required for 200 instructions, and shall be removed prior to the final assembly.

21. To further facilitate the testing operation, the techniques described below may be used, *if* the program does not require the use of all of memory.
    - *a.* Blank exit points may be inserted into the program at various locations to facilitate the addition of instructions during testing. These exit points will generally take the form of a NO-OP (or NOP) with

an operand of 00000, so that a JUMP, BRANCH or TRANSFER instruction can be inserted into that "reserved" location. If additional instructions are required in the routine in which the NO-OP exists, a jump can be made to upper memory without destroying any of the instructions in the routine. In order to facilitate the removal of these testing aids, a specified comments column of the symbolic entry card should be punched with an identifying punch, so that the symbolic deck can be sorted on this punch and the testing aids removed automatically before the final assembly. (The testing aid instructions should never be labeled, if they are to be removed.)

b. In order to assist in the rapid review and identification of memory dumps, special alphabetic constants can be coded in various routines which will identify the routine on the memory dump. Thus, the program number or name can be loaded into a specified place in memory, so that the memory dump will be permanently identified. Similarly, each of the program blocks can be preceded with a constant phrase which contains the block number or letter; this phrase will then appear on the memory dump immediately preceding the pertinent instructions. Again, an identifying punch in the comments card may be used to remove all of these aids prior to the final assembly.

22. All machine language correction cards shall be punched as follows:

a. They shall be punched on a card with a different corner cut from the condensed program deck.

b. They shall carry the following identifying information
program number and phase
date of correction (to reference the test results summary)
page number of the symbolic listing where correction is recorded

23. Temporary corrections may be included if deemed necessary. These corrections may be used to stop the system at appropriate locations prior to dumping memory, or to suppress checking or controlling on information not available in the test data. Such patches should always be punched on cards of a different color to insure their removal prior to final testing.

24. If a correction is made in error, or a correction must be further corrected, the original must be removed and replaced. A correction may not be made "on top of" another correction; reshuffling of the card deck will completely destroy the intent of the multiple correction.

25. If a logical error is found a correction must be made to the block diagram at the same time that the correction is made to the program. The block diagram must be maintained current during testing.

Production and Systems Testing

26. After all of the prepared test data is processed accurately by the program, the program is ready for production testing.

27. Production testing shall be performed the same way in which the program will ultimately operate. There shall be no temporary corrections in the program. All of the input information shall be "live" data.

28. The amount of test data used in production testing depends on the frequency with which the program is designed to operate. If daily, the

production test must cover at least two days' work. If the normal run cycle is monthly or less frequent, only one period's sample data is required.

29. The systems analyst who designed the system is responsible for checking output of the production test of each program. If the program is accepted by the analyst as producing satisfactory data the program is deemed ready for a systems test.

30. The systems test shall be operated similarly to the production test, on approximately the same volume of data. All of the programs of the system must pass the production test stage before the systems test.

31. The final outputs of the systems test shall be evaluated by the systems analyst and the user who will ultimately use the information. The systems test should include a run-through of the entire system, with false end-of-quarter and end-of-year conditions simulated, if necessary.

32. Upon agreement by the systems analyst, the programming department and the user, a parallel run shall be initiated.

33. For programs with a run frequency of between one day and one month, the parallel run shall operate for a minimum of one weekly cycle and a maximum of two months, before the old system is discontinued. If the parallel run is unsuccessful the new system shall be temporarily discontinued. If the parallel run produces accurate and suitable output, the old system may be discontinued provided that control personnel continues to evaluate all output data. Any run which has not been paralleled after the old system has been discontinued must be run under complete control of the using department for the first two times.

Conversion

34. Whenever possible a modification of an operating program shall be used to convert the data into a format acceptable to the new system.

35. If it is necessary to create special conversion programs such programs will be written under the standards applied to normal production programs, with the following exceptions:

The only documentation required is a macro-logic chart and operator's instructions.

Conversion programs need not be tested by the systems analyst; live data may be used for all tests and no parallel run is required.

Testing of Documentation

36. In order to validate the accuracy of the documentation (prepared in accordance with the standards outlined later in this Chapter) it shall be tested as follows:

a. After completion of the documentation the operator's instructions shall be turned over to an operator who has *never* operated the program before.

b. Using the test data prepared for the production test, and including some test data which will cause programmed stops, the operator shall operate the program, without assistance, following the exact instructions in the manual.

c. If the operation is successful, the documentation is assumed to have been understood by the operator and is considered fully tested.
d. The documentation shall be read and edited by the programming supervisor, who shall determine the adequacy of the instructions and explanations.

"OPEN SHOP" VERSUS "CLOSED SHOP" TESTING

At the present time, many smaller installations are allowing the programmer to assist the operator, and in many cases actually operate the machine during program testing. Conversely, other installations never allow their programmers to go near the machine, and all programs are tested by the operating group on a 24-hour or better turnaround basis. A number of arguments can be made for either system, and proponents of both methods will fight violently for their opinions. The following arguments apply in favor of "open shop" testing, i.e., allowing the programmer to operate his own tests:

1. The programmer can find many more errors in one shot. After hang-up on a specific condition, the programmer may directly recognize and correct the error or he may be able to jump around the error, after dumping memory, and thus catch more than one error with each shot on the machine.
2. If present, he may detect and correct operational weaknesses (instances in which programming can improve operation) in his program.
3. Programmers are often better machine operators than the assigned operators, and would make less operating errors during testing.
4. Total program testing time is reduced because there are no long delays while waiting for the output of the "remote" test.

Similar arguments may be given in favor of closed shop testing:

1. Programmers often foul up their program by playing with the console after a hang-up.
2. Programmers waste a great deal of time, before and after testing, in disrupting other work to get on and off the machine and in waiting for machine availability. During direct testing programmers rarely do any other work.
3. A great deal of machine time is wasted in allowing programmers to do machine testing.
4. If the programmer must write test instructions for operators, he organizes test shots better and learns to document programs better.

Having seen both systems extensively used, the author recommends the

following procedure as the most economical, in consideration of overall management objectives:

a. All programmers should be given a minimum of two weeks of on-the-job operator training, by acting as operators within the installation for that period of time. This will familiarize the programmer with operational requirements, and will enable him to translate his understanding of operator problems into meaningful, coordinated programs and programming manuals.
b. All program testing should be done according to the "closed-shop" principle, having the operator run the test, under explicit instructions from the programmer. The programmer must prepare complete instructions and in turn be guaranteed a maximum turnaround time of 8 hours or less (depending on the installation schedule).
c. An operating priority system should be established providing for at least three priority levels: priority production runs, normal testing and normal production. "Crash" programs should be given priority in testing and guaranteed a maximum turnaround of two hours.
d. All programmers should be allowed to observe their test runs if their schedule permits. All programmers should be *required* to observe the first production test of their program.
e. When program testing precedes installation it must still be done by operators at the test center. In this case, since the operators are not yet highly trained, the programmer should accompany the operator during the first few weeks of testing.

STANDARD TECHNIQUES AND SUBROUTINES

In larger installations the term "standards" often describes only the standard techniques and subroutines which have been developed within the installation to assist programmers. In the initial development work a great deal of value can be obtained by creating subroutines usable in many of the programs to be developed. These may include:

- Multiply and divide, where not provided in the hardware
- Printer line-up
- Date card entry and encoding
- Disc file randomizing
- Tape and item control
- Error analysis
- Format control and edit
- Data generation
- Utility routines to assist in operations and testing such as
  transfer trace
  memory snap print
  tape duplication

testing monitor
operations monitor
debugging aids.

These routines may be incorporated as "macro-instructions" or poly-operations on a library tape, or they may exist in memory at all times, or they may exist in card form to be assembled with each program that requires their use. In either case, provision must be made for

- The contribution of these to the installation library
- The dissemination of these to all programmers
- The documentation in a standard format, to avoid repetition in each programming manual.

To encourage the contribution of such standard techniques by programmers, a simple "suggestion" type form may be used. Each programmer who wishes to generate a "macro-statement" can fill out such a form and thus have the coding introduced on the library tape. To make the information useful to other programmers, an indication should be given of the function, purpose, memory requirements and utility of the routine. A copy of the routine can then be incorporated in the manual of standards of the installation. Figure 5-6 shows a form of this type.

Documentation of the subroutine or macro-statement requires the following:

- Name of the subroutine
- Calling sequence to use the routine
- Purpose of the routine
- Programming requirements
  - entry parameters
  - exit parameters
  - execution time
  - memory size
  - restrictions and limitations
  - other subroutines referred to or used
- Operating procedures and halts, as applicable
- Logic block diagrams
- Test data
- Sample or illustration of usage

M-II

Standard Technique

Programmer_____

Date_____ No._____

Purpose or function:

Programs in which used or usable:

Label	Operator	Operands and Notes

Supervisor Comments	1	2
Approved_____		Approved _____

Fig. 5-6. Standard Technique. (*Courtesy*, The Bowery Savings Bank)

DOCUMENTATION STANDARDS

By far the most important, the standards for program documentation specify the extent to which the programmer should support his efforts in writing. After testing is completed, the program will work, whether it is documented or not; as a result there have been too many instances where the documentation has been completely neglected, or written in a haphazard manner. Documentation is a vital part of the requirements of a computer installation; its importance and necessity cannot be overstated.

Reasons for Documentation

The most commonly stated reason for good documentation is that it prevents the individual programmer from becoming indispensable. Management is normally quite concerned about the status of its programs whenever programmer turnover exists. Because of the current personnel requirements the turnover rate in the average installation is quite high (ranging from 20 to 50%); this results in emphasis being placed on documentation by management. More than this, a trained programmer represents a valuable asset to a company, especially in the subject area in which he has done the bulk of the work; in this instance he is bound to know a great deal about the methods and requirements of this application area. As a result he becomes quite valuable in the actual operation of the using department, *provided* he can be promoted without causing problems in the data processing operation.

The real costs of no documentation become apparent when programmers leave the installation. There are three areas where difficulty will be encountered:

- In completing a program in the development process. In many instances management has had to decide to scrap the work already completed and start over again, primarily because the work already completed is difficult to understand and the time required to thoroughly analyze the existing parts would be greater than the time needed to rewrite the entire program
- In making changes to programs currently operational. It should not be difficult to make simple changes to such programs; nevertheless, the complexities of the program, the changes which may already be in existence and which have not been properly documented, and the fact that simple changes may have a major effect on other program sections, or related programs, often make it an extremely difficult task. After a programmer leaves the cost of changing his programs often increases by a factor of ten or more.
- In converting programs to a new machine. No matter how advanced the computer installation, sooner or later the equipment becomes obsolete. To replace the equipment most of the programs or systems will have to be rewritten or converted in some manner. This task becomes extremely difficult in the absence of complete and up-to-date documentation.

There are several other reasons why documentation is necessary:

- It serves as a record of the work performed; program documentation is frequently the only evidence which management has of the expense of thousands of dollars for computer installation.
- It increases the efficiency of the entire installation; both programming and machine operation are placed on a better organized and disciplined basis if proper documentation is required and supplied. Direct savings in machine operation costs are also attainable through the proper planning and sequencing of the operating steps.
- It provides the users of the program with a basic understanding of what is being provided and of the type of formulas, controls, etc. that are included. It also acts as protection for the data processing department by preventing misunderstanding about the purposes and outputs of the program.

Users of Documentation

There are four basic users of the documentation supplied with the program:

Management.—Management requires program documentation for purposes of review and for general information; management is normally not concerned with the technical detail; their major requirement is for an understandable general description of the program's function. This will enable management to understand the system concepts without having to learn the details of the program.

Program Users.—In a commercial installation the users of the program should generally be familiar with the output, its meaning, its derivation and its accuracy. These users represent the operating departments of a company, and their concern is also with the general description, and with a sample output, or output layout. It is their function to determine initially if a program is suitable, or if changes are desired to reflect changes in the business operation.

In a scientific or engineering installation, the users are generally the scientists or engineers. They must be thoroughly familiar with all of the available programs in order to determine whether any will fulfill particular problem requirements. A mathematical abstract should therefore be provided that indicates the program functions, the formulas involved, and the previous users. The engineer can then determine, without a detailed understanding of programming, whether this program suits his purpose, or can be modified with little effort to provide him with

the necessary information. Since engineering use is non-cyclical, it is necessary for each engineer to make up his own input cases, and a detailed layout of input requirements should therefore be included in the documentation.

Operations—Production—Set-up.—In a commercial installation, the operating function is cyclical and is delegated to a separate operations group. In an engineering installation it is common to establish a separate group of "set-up" or "production" programmers, whose function is to provide the program deck, its documentation, and valid input information to the operations group before use. This intermediate group also acts as liaison with the engineers, and therefore requires exact information about each program. In both cases, the operations group requires information for normal operation, and for the rapid correction of errors. Part of the documentation must be meaningful and concise operating instructions in a rapidly accessible format.

Programming.—In order to make desired changes or to convert to other equipment, the programming group will be the most demanding user of program documentation. The programmer assigned to change a program he did not write will want to assimilate all available information about the program as rapidly as possible, getting right to the heart of the section where his change has to be made. Programming documentation should therefore include both detailed descriptions and assistance to future programmers in making those changes that can be partially anticipated.

Types of Documentation

Many kinds of documentation are required. Chapter III discussed the job specification and the flowcharts necessary. Their relationship is shown in the "organization" chart of documentation shown as Figure 5-7.

This section concerns itself primarily with program documentation, i.e., the programming or maintenance manual and the operating instructions. There will be differences in the organization of documentation from one installation to another; an engineering installation rarely has programmed halts, for example, and a real-time system will never have programmed halts. Nonetheless, for purposes of illustration all of the required documentation has been shown for the most extreme installation.

Documentation Contents

The following pages describe in general terms the contents of the manuals which should exist for each program. Minimum standards should be established and rigidly enforced for each installation.

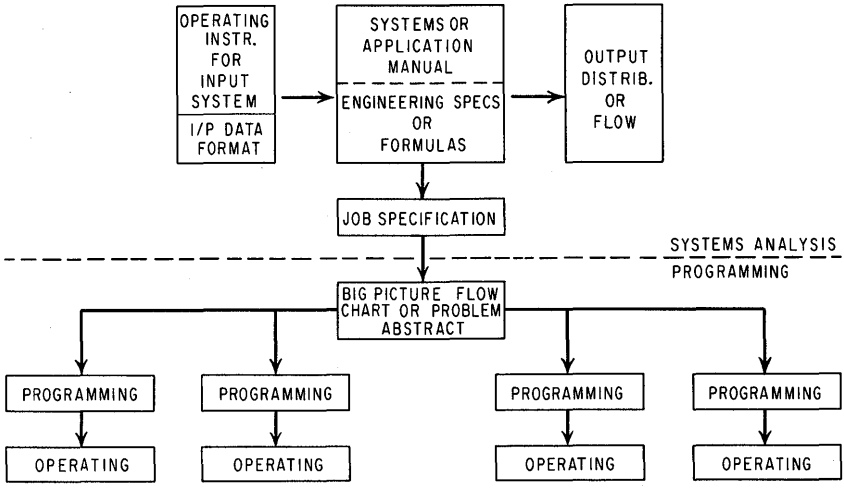


Fig. 5-7. Documentation Hierarchy.

General Rules.—The general rules listed below apply to all documentation or manuals produced by an installation.

1. All documentation shall be provided with a permanent-type, cardboard or leatherette cover, with a suitable label to indicate the manual name, number, date and revision.

Since documentation should be respected by the user, it should be given the necessary authenticity. A cover is one means of achieving this; the permanence of a cover is desirable if it is considered that the modest cost of a cover is far less than the cost of the labor required to produce the manual.

2. All documentation shall be given a title page, which shall include the program name, the program number, the programmer name, the date and revision number, and the copy number, if more than one copy is prepared.
3. All documentation shall have a revision page. This page shall show the revision number for each change, the affected sections, the date of the revision and the name of the person who made the change and updated the manual. [Figure 5-8 illustrates a sample revision page.]
4. All documentation shall have a Table of Contents.

The creation of a Table of Contents takes very little time; its utility in providing rapid access to the information is obvious. Tables of Contents are illustrated as a part of the Sample Programming Manual.

REVISION PAGE

DATE	BY	SEC	PAGE	REVISION	NO.
8/1/62	O.L.T.	II A	14	Added Halt # 0042 - DATE CHECK	1
11/15/62	RI	III	28	Changed logic of "get" routine to handle blocking of 10 - rather than 8	2
4/1/63	Proj. Supvr.	III V	18 19 31 32 39	Revised O/P layout To include added state tax: revised logic to calc. tax; added field to master file	3
4/19/63	" "	" "	" ALSO 33A,B Oper. Instr. Sheet	created monthly O/P tape for tax reports, after new ruling; used alt. run # new function Revised oper. instructions	4

Fig. 5-8. Revision Page.

Operating Manual

The maintenance manual consists of two sections: the operating manual, used for machine operation and set-up, and the programming manual, used to make changes to the program. The entire manual is the basic documentation for the program. It is not necessary to give the machine operator the entire manual; his interest is in the sections outlined below.

Abstract or General Description.—The first section of the manual should be a brief general description of the program, its function, its features, its options, and its basic inputs and outputs. It should describe the program in layman's language so that the operator will understand each phase and properly distribute the outputs to the next functions. It should be brief, perhaps limited to one typed page, but should include all of the desired elements. For engineering programs, the general description should include a basic abstract and the formulas solved by the program. The method of solution, if it can be simply described, may also be a part of the general description; if it becomes too cumbersome this should be referred to the detailed description, which will not be a part of the operating manual.

Flowchart.—The next section of the manual should be a flowchart; this flowchart has a multiple purpose:

- It defines the input and output
- It establishes the timing for a sample run, to assist in scheduling and program testing

- It provides a brief synopsis of the program functions
- It lists the minimum configuration on which the program can be run.

The minimum configuration is sometimes maintained on a separate page. It should not be neglected. It is very common to see installations shut down entirely even though only part of the equipment is inoperative. However, many programs do not require all of the units or hardware features. Further, it may be possible to make a simple modification to the program, or alter a switch setting or parameter card, reducing the equipment needed by crippling a non-essential function such as tape alternating.

A sample flowchart is included as Figure 3-14*a* and 3-14*b*, page 62.

Operating Instructions.—It is common to see a "crackerjack" programmer spend hours in refining a subroutine to save a few microseconds. But the same programmer may be seen writing his operating instructions on a piece of scrap paper without any regard for the fantastic waste of machine time thereby created. For example, by not specifying the *sequence* in which the machine uses the input-output units at the beginning of the program, he fails to provide the maximum overlap of operations and set-up. As a result the operator may set up the alternate tape, for an output that will not be produced in the first 10 minutes, before loading the program. At least two minutes is thus wasted, during which the program could have been in full operation. Without the proper documentation the operator will never know the sequence in which the machine should be set up for maximum overlap.

Operating instructions, to be effective, should be divided into several sections:

Overall Console Set-Up Summary Page.—The set-up summary should provide the operator with a check list of all of the required inputs and outputs. It will specify the printer forms, the carriage tape, the types of input cards, their quantity and sequence, and provide the complete set-up of all console switches. A sample set-up summary is illustrated as Figure 5-9*a*.

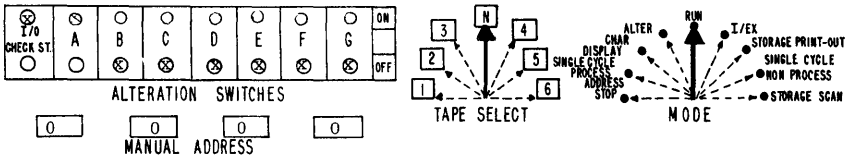
Set-Up Instructions.—Following the set-up summary should be a detailed list of *sequenced* set-up instructions. It is the programmer's responsibility to tell the operator in which sequence the machine uses its components. He should thereby specify the exact sequence in which the machine is to be set-up.

Normal Operating Notes.—The messages and halts which will occur should not come as a surprise to the operator. In order to operate as efficiently as possible, the operator should know exactly what to expect from the normal operation of the machine. This will also enable him

Electronic Data Processing Division

II -OPERATORS INSTRUCTION SHEET-1401-

PROG. # _____ PROG. NAME Demand Deposit - Daily Transaction Merge



READ FEED				
NO. OF CARDS	FILE NAME (80)		SEQ.	STACKER #
300	Merge Program		1	NR
10-30	Ledger Acct. Control Cards	T	2	NR
1	Date Header Card	H	3	2
1	Input File Control Card	F	4	2
xxx	Input Data Detail Cards		5	1

PUNCH FEED		
FILE DESCRIPTION	SEQ.	STACKER #
80-80 Invalid Input Card (Use blank cards)		NP

PRINTER					
SEQ.	FORM #	FORM NAME	CARRIAGE ALIGNM.	NO. COPIES	CAR. TAPE
1		14 1/2" x 11" stock	201-332	1	Spcl

TAPE UNITS DESIGNATION								
TAPE UNIT NO.	FILE NAME	I O W	PROTECT RING IN	PROTECT RING OUT	FILE DESCRIPTION OR COMMENTS	RECORD LENGTH CHAR	RETENTION CYCLE DAYS	NEXT USE
1	Sorted Input File 1	Input		x	Fixed Block 20 Rec.	26	5	None
2	Sorted Input File 2	Input		x	Fixed Block 20 Rec.	26	5	None
3	Sorted Input File 3	Input		x	Fixed Block 20 Rec.	26	5	None
4	Merged Transactions	Output	x		Fixed Block 20 Rec.	26	5	Posting Run
5	Short List	Output	x		Variable Block/50 Rec.	10	5	Short List
6	Acct. Overflow	Work	x		Variable Block	26	5	None

OPTIONS OR MODIFICATIONS			
NO.	DESCRIPTION	FUNCTION	BY *

Fig. 5-9*a*. Operator's Instruction Summary. (*Courtesy*, The Diebold Group, Inc.)

to rapidly recognize any abnormal conditions, to take the required action with the minimum waste of time. The normal operating notes should include the halts and the messages which should be expected, and whatever action the program will take to signal the end-of-job condition.

Take-Down Instructions.—Most programmers assume that the operators will be able to remove the ouput and input files correctly but it is important to specify the sequence in which this should be performed. For example, a tape file may be completed in the middle of a program. Its removal may be overlapped with the completion of the run, and more important, the drive may be used to set up the next sequential problem. Take down instructions should therefore not be ignored; their value is substantial relative to the cost of their preparation.

A sample sequence of operating instructions is shown as Figures 5-9*b* and 5-9*c*.

Abnormal Conditions.—The occurrence of an error may be assumed to be an abnormal condition. When it occurs, the program generally communicates with the operator by typing or printing a message; frequent repetition of the condition may warrant operator intervention. If the error is correctable a procedure for this should be documented; if it is not the operator should be instructed to call his supervisor, the programmer, or the customer engineer, dependent on the cause of the error.

Messages.—A listing of the messages should be provided for each program. This list should include the normal messages separately from the abnormal messages. If a message is followed by a program halt, the halt number should be referenced in the message. The cause of the message should be explained, and if a halt occurs at the same time, the operator action should be indicated. Since halts are documented separately from the messages, the page number of the halt should be indicated in the message explanation:

TTC RD TP 20X: A tape transmission check has occurred on tape 20X. If it cannot be corrected by the program after nine tries, Halt 124x will occur. (See page *XXX* for a description of the action to be taken at that time). If the error is corrected the message will not re-occur until another TTC is found. If corrected TTC's occur on the same drive more than ten times continue the program, but advise the Customer Engineer. At the earliest opportunity replace the file on another drive on the same channel.

Halts.—Program halts should be documented extremely carefully. An operator error at the time a halt occurs may be misconstrued or misinterpreted as a machine or program failure. As a result, the operator should be instructed, in detail, as to what action is required of him.

Programmed halts should be kept to an absolute minimum, since the program should be able to make most decisions for the operator. Ideally, there should be no halts, except verification halts for printer

II. OPERATING INSTRUCTIONS

A. Initial System Setup

1. 1402 Card Read/Punch - Turn ON the read and punch. Place the "Daily Transaction Merge" program in the read feed. The last cards of this deck are the ledger table control cards. The last of these cards is punched with nines in columns 1-9 and a T in column 80.

Place the two current control cards behind the program in the reader. The first control card is the (H in 80) Date Header Card; the second is the (F in 80) Files Control Card.

2. Set up the CONSOLE function and sense switches as shown on the Operators Control Sheet. Then depress the Check Reset and Start Reset keys. Depress the Load key on the 1402 Read Unit and the program will start loading.

3. 1403 Printer - Mount the "Daily Transaction Merge" carriage tape. Mount one part 14 1/2" x 11" stock form.

4. Magnetic Tape Files - Mount the "Sorted Daily Transaction" input tape files on tape units 1, 2 and 3. Remove the file protection rings from these files before mounting. When using 2 files - mount only tape units 1 and 2, dial 3 off. When using 1 file - mount only tape unit 1, dial 2 and 3 off. Mount "PAL" system work tapes on tape units 4, 5 and 6. Set the file protection rings in these files before mounting. Check the external labels on these reels to insure that they are available for use as output tapes. Place new external labels on these reels identifying their new contents and today's date.

Tape unit 4 - "Daily Transaction Merge - Merged Output"
Tape unit 5 - "Daily Transaction Merge - Short List Output"
Tape unit 6 - "Work Tape"

5. 1402 Card Read/Punch - Place the detail card input (if any) in the read feel. Fill the punch feed with blank stock cards. Then depress the 1402 Start key to start processing.

B. Operating Notes

1. The program begins by initializing itself and reading in the control cards. If any of these cards are invalid a programmed halt will occur. All of these halts are in the 600 series. If any of these halts occur, the program will have to be reloaded after correcting the invalid control card.

2. Halt H701 is a printer check halt. The printer will have printed the following data: the number of input tape files, the current tape date and the current report date. The printer has skipped the form to channel 1 after printing.

If start is pushed, the same data will be printed again, the form will again skip to channel 1, and H701 will reoccur.

Check the printer alignment (all data must be printed on the forms, and the skip to channel 1 must have occured properly). Check the printed dates for accuracy. Check the number of tape input files printed. When satisfactory, set sense switch B on and push the start key to enter the main program.

3. The only other programmed halt which should occur is H899, end of job. When this halt occurs, the message "End of Job" will have been printed on the last page of the report, and all of the tape files will be rewinding.

If any other programmed halts occur, an error or exception condition exists, and the manual should be consulted for the corrective action to be taken.

C. Completing the Program

After Halt 899 (End of Job), the operator should remove all materials as follows:

1. Magnetic Tape Files

a. Dismount the input tape files from tape units 1, 2 and 3. Each must have an external label indicating "Daily sorted transaction input" and today's date. These tapes are to be preserved for five days and then will become available as "PAL" system output tapes.

b. Dismount the output tape files from tape units 4, 5 and 6. Immediately remove the file protection rings from these tapes.

Each must be properly externally labeled as indicated in the Initial System Setup section.

Tape unit 4 - "Merged Output" will be used on the "Daily Posting Run" program.
Tape unit 5 - "Short List" will be used on the "Daily Short List" program
Tape unit 6 - "work tape" is of not further value to the daily operations, and is immediately available as a "PAL" system output tape.

2. 1403 Printer

Remove the listing and hand-label as the "Daily Transaction Merge Journal and Ledger Balance" report.

Remove the carriage tape and place it on the 1403 carriage tape rack.

3. 1402 Card Read/Punch

Normal Punch Pocket. Any cards in this pocket are copies of invalid input cards. Manually label, date and file as "error inputs - Daily Transaction Merge".

Normal Read Pocket - the Merge Program. File in the "Daily PAL Program" file.

Read Pocket 1. Label and file as "Daily Transaction Merge - Detail Card Inputs".

Read Pocket 2. These are the date and file control cards. These cards may be destroyed.

Fig. 5-9*b*. Operating Instructions. (*Courtesy*, The Diebold Group, Inc.)

V. OPERATING INSTRUCTIONS (cont.)

B. Set Up Instructions

1. Mount Deposit Accounting Master File (most current date) on servo unit No. 1.
2. Place 3-ply stock paper in the printer.
3. Request the program.
4. Place parameter cards (4) in the card reader, followed by one or more sentinel cards.
5. Place blank cards in the card punch (abandoned account control cards - form MIDCO 9525).
6. Initiate operation.

   Normally, two initial type-outs will occur:

   a) Type out of parameter information:

      5 YR △ 06XX △ 6 YR △ 06YY △ 10 YR △ 07 ZZ

      PARAM OK ANS YES OR NO

   b) Print out of headers for line up, on the printer.

      PRINTER OK ANS YES OR NO

   Following their acceptance, the program will proceed until end of job:

      END OF JOB D01A

7. Take-Down Procedure:

   a) Remove the listing from the printer.
   b) Remove the 5 year cards from card punch stacker 0.
   c) Remove the ten year cards from card punch stacker 1.
   d) Remove and save the type out and parameter cards.
   e) Remove the master file from Servo #1.

Fig. 5-9c. Operating Instructions. (*Courtesy*, The Bowery Savings Bank)

line-up or tape label checking. Nonetheless there are occurrences of machine errors, where the customer engineer should be called, or data errors where the data coordinator should assist. In these cases, the options which are available should be carefully outlined:

Depress the start key to try the same operation again. Depress the reset key followed by the start key to enter the Automatic Restart Procedure.

Correct the apparent input error, turn ON alteration switch 15, and place the card in error in the read feed, followed by the remaining cards. Depress the start key to read again.

If the input item is to be deleted from the program, turn OFF alteration switch 15. Depress the reset key, followed by the start key; the input item will be typed out followed by Halt 2301. At this point, turn ON alteration switch 15 and depress the start key to delete the item.

In documenting programmed halts, avoid the use of phrases which have limited meaning; these can be extremely frustrating. For example saying: "This halt should NEVER occur" does not assist the operation after it has occurred. The phrase ". . . something has gone wrong with the program . . ." would also appear to be somewhat redundant to the operator who is trying to get his job done. Restrict the use of humor, especially when it is so subtle that it is confusing; do not type poetic messages where meaning is lost and try to retain a perspective on the situation that exists when the halt is reached in the actual operating atmosphere!

The inclusion of a programmed halt should be carefully considered. If it is included in the program it should be equally carefully documented, using a separate page for each halt. (Figure 5-10 illustrates a form used for halt documentation). If more than three halts are included as a part of a program a separate index of halts is extremely useful to the operator in trying to locate the halt in the manual. For example,

Halt No.	Cause	Page
0112	Printer line-up	19
2114	Recurring tape error	23
2234	Tape label error	20

Console Alteration—Sense Switches—Alter Keys.—Most computers provide interrogatable console switches that may be used to alter the course of the program or specify the use of options. Since these switches or keys can accidentally be left in the wrong position, the program should generally tell the operator which options it is selecting before proceeding. In any case, the documentation should clearly state the purpose of each switch, and its effect on any section of the program, since it could be turned ON accidentally during program operation. In general, switches should be interrogated only during the initial functions of the program; the effect of their use should then be prevented except when an operator decision is required. Nonetheless, complete documentation of both sides of the switch function is a definite part of the operator's instructions, over and above their status as indicated on the set-up summary. For example:

Electronic Data Processing Division

PROGRAMMED HALTS - 1401

Page___of___

Program No._____ Program Name_____

Halt Description_____

Halt No.	"A" Register	"B" Register	"I" Register

Message	None

CAUSE:

CORRECTIVE ACTION AND EFFECT:

OPTIONS &/OR MODIFICATIONS*:

*INITIAL ALL MODIFICATIONS.

Fig. 5-10. Programmed Halts. (*Courtesy*, The Diebold Group, Inc.)

Alter Key	Status	Section	Effect
#1	ON	Initial	Will repeat the printer line-up routine
	OFF	"	Will assume correct line up
	ON	Running	If tape error—will try reading again (See halt 124n for complete instructions)

Operating Manual—Layouts.—The operating section should be provided with the layouts of the input cards and the output forms. This will enable rapid location of input and output errors and prevent the use of the wrong card file or the wrong report form. It is not necessary

to include tape records or memory layouts since these would serve no useful purpose to the operator.

Sample reports and sample card forms should be included wherever possible to further assist the operator. The last layout in the operating manual, which is most frequently forgotten, is the layout of the carriage tape for the printer, if a special tape is required. It will not be possible to reconstruct the carriage tape layout after the tape has been destroyed.

Programming Manual

The above information represents that part of the maintenance manual required for effective machine operation. The remainder of the manual is required for proper program maintenance and for general familiarization with the detailed program functions.

Since the operating manual layouts were the last items in the previous section, the next item will logically be the remaining layouts required for the program. The sequence in the manual is not terribly important; if the material is required a logical sequence is desirable. The most important factor in the sequence is that each and every manual should have its contents arranged in the *same* sequence; this will assist greatly in locating the desired information.

Programming Manual—Layout. Memory Layout.—An approximate layout of memory should be included to explain the usage the program makes of the available facilities. For example:

From	*To*	*Used for*
00001	00600	Initialization Program
00600	02400	Main Program
02400	02900	Constants
04000	05000 *	Table of arguments
05001	06000 *	Input area

* All addresses marked with an * are exact, and are used in address arithmetic; These should not be altered under any circumstances, except as indicated in the section on modification, page *XXX*.

Tape Layouts.—Tape layouts should be included for each tape used in the program, unless a central record of all tape layout files is maintained. In the latter case, specific references to the central file should be made to indicate the particular file used for each input and output.

Tables.—Tables or data arrays should be carefully documented, since they are apt to be subject to considerable change. If no method is programmed for the updating of the table, some indication should be given in the table documentation of the sequencing and of the controls on which the program relies. A part of the table documentation should be

a sample indication of how the table may be changed by deletion, by addition, or by replacement of a data item.

Detailed Description.—Perhaps the most important element of the program documentation is the standard detailed description of the program and its functions. This should describe in detail the functions of *each* of the logical blocks or segments that constitute the total program. If a standard has been established defining the program blocks (e.g. no more than 26 blocks per program) the detailed descriptions can tie directly into the macro block diagram as follows:

Block F—"Get" Subroutine for card input.

Control is transferred to Block F by the output control routine Block G, after a card input record has been merged, and a new card is required. The routine has the following functions:

a. A card is read and stored in the "card input area" AR1.
b. The input card is checked for sequence, against the previous input card. The card is further checked to make sure that all fields that should be numeric and fully punched are correct. A card that fails either test is printed and punched out, and the following card is read. If more than 40 invalid cards are detected by the routine, the program will enter a terminal halt (0966), the input cards must be corrected and the program restarted from the beginning.
c. Cards which pass the input tests are printed "four-up," for reference only, complete with generated overflow page headings. After reading a valid card, and printing on every fourth, control is transferred to Block B—Merge. . . .

If this kind of description is generated for each and every block of the program, it is very easy to become familiar with the various elements which make up the processing function, for purposes of making changes.

All standard subroutines, SHARE codes, or other utility sections documented elsewhere should be referenced in this section and its documentation indicated.

Macro-Block Diagram. The one-page block diagram, described in Chapter III, should be included as part of the basic program documentation.

Micro-Block Diagram.—The micro-block diagrams which further describe the logic inherent in each of the blocks or subroutines are always included as a part of the programming manual.

Special Lists of Helpful Information.—In order to assist other programmers in making changes or conversions, the documentation should provide various items in list form, including:

- Electronic switches used by the program
- Counters or accumulators used
- Special constants and their designations
- Buffer areas and other significant factors

These lists should specify the symbolic tag of the item, its significance, and the location(s) where the item is used, referenced or changed. A sample list is shown as Figure 5-11*a* and 5-11*b*.

Features, Cautions and Modifications.—A section should be included in each manual which describes special features included in the program. This should include the use of special techniques, the derivation or method used to program the formulas, areas for future changes or improvements, cautions about tight routines, and perhaps the areas where modifications must be made periodically. If a calculation depends on a percentage which may be changed by government regulation (as the tax percentage, or the F.I.C.A.) or through some other external influence, the constant should be kept separate and distinct, and the method of

VII. TABLE OF PROGRAM SWITCHES

Symbolic Tag	Switch Name	Type	Initial Status	Reverse Status on Condition & Use
CX2A	No input tape 2	WM	OFF	ON for no input tape 2 or 3.
CX3A	No input tape 3	WM	OFF	ON for no input tape 3. (merge control)
CX1C	End of Job	SIGN	+/OFF	-/ON for end of job, after last ledger print out, enter EOJ routine.
CX1F	Error card	WM	OFF	ON for an invalid input card. Causes an error card print-out and return to read the next card.
CX1G	Account with checks	WM	OFF	ON for a check detail trans-action.
CX2G	Package Post	WM	OFF	ON at the beginning of the High Volume Ledgers.
CX3G	Overflow end	SIGN	+/ON	-/OFF-set off after process-ing the last overflow record in an account.
CX4G	Start Short List	WM	ON	OFF-set off after processing the first record of a Short List, set ON after complet-ing the write out of the entire Short List batch. (up to 200 rec)
CX5G	Write tape switch	WM	OFF	ON-set on at the end of an account to force the write out of a Short List or Overflow tape block.

(Continued)

Fig. 5-11*a*. Table of Program Switches. (*Courtesy*, The Diebold Group, Inc.)

IX. PROGRAM ELEMENTS (Cont.)

Relative Location	Label	Use
	U (D01AKP01)	Six Year Date in format 000YYYYYYYYMMMM Year in BCD, Month in Octal
	L (D01AKP01)	Five Year Date in above format
	L (D01AKP02)	Ten Year Date in above format
	CPWS	Temporary Storage - Console Replies
	AMWS1	Temporary Storage-Trailer Search

C. Counters

Relative Location	Label	Use
	D01AEB01 - 9 Words	Balance Before Dividend Totals By Branch
	D01AEB02 - 9 Words	Dividend Totals By Branch
	D01AEB03 - 9 Words	Balance Totals By Branch
	D01AEB04 - 9 Words	Item Counts By Branch

Note: Each of above sets of counters are addressed by direct incrementation of branch number to base label. Base label locations are used for grand total accumulations of respective totals.

	D01AP007 - 8 Words	Page numbers by Branch (Base + BR# - 1 = Page No. Brn).

D. BUFFERS

Relative Location	Label	Use
	D01ATI04 - 504 Words	Master File Input
	D01AP001 - 17 Words	Detail Print Area # 1
	D01AP002 - 17 Words	Detail Print Area # 2
	D01AP003 - 20 Words	Page Header #1
	D01AP004 - 17 Words	Page Header #2
	D01AP005 - 17 Words	Page Header #3
	D01AC001A- 17 Words	BCW and Buffer #1 Five Year
	D01AC001B- 17 Words	BCW and Buffer #2 Five Year
	D01AC002A- 17 Words	BCW and Buffer #1 Ten Year
	D01AC002B- 17 Words	BCW and Buffer #2 Ten Year
	CPCI - 17 Words	BCW and Buffer for parameter card input
	AMWS2 - 130 Words	Account record working storage for Trailer Search

Fig. 5-11*b*. Program Elements. (*Courtesy*, The Bowery Savings Bank)

change should be clearly indicated, so that a sudden change request does not cause major problems.

This section should also be used to boast a little about the many techniques or "tricks" which are included in the program, so that program change can be attempted without excessive danger. Programmers enjoy talking about their feats, and their programs; if this same energy is used to set the information down, it may be of great assistance in future implementation.

Assembly Listing.—Part of the basic documentation of the program is the latest assembly listing, carefully marked with the date of assembly and the revision number.

Memory Dump.—To assist in correction of latent errors, a core load dump or a dump at end of job should be provided as a part of the basic documentation. This could prevent the need to run the program in case of error; it may enable an immediate assessment of the corrections included in machine language, and it may provide a comparison to the memory dump taken by the console operator at the time of failure.

Test Data.—Part of the basic documentation is the test data used to make the final determination that the program is operational. This test data should be maintained with all changes, and *all* such data should be used to test the program after any change. This will insure that the change has not accidentally affected other sections of the program.

The Sample Manual

The development of the necessary standards to prescribe the required documentation should be implemented as a part of the standards manual. To assist the staff in the development of the appropriate documentation, and to provide a clear illustration of the techniques and organization of the program manual, a sample manual should be included, or referenced, as a distinct part of the standards manual. In either case, the table of contents of a sample programming manual should be displayed in the manual of standards. Two such tables are shown as Figures 5-12*a* and 5-12*b*.

A sample manual can be created by initially documenting a program that is considered "average," i.e., not overly complex, and not overly simple. By creating the best possible documentation for this program, and by giving each programmer a copy of this complete material, an effective documentation standard has been established. All subsequent documentation should be qualitatively and quantitatively measured against this manual.

PROGRAM CHANGE ADMINISTRATION

The most serious problem faced by an operating installation is the accurate maintenance of all programs used in the operation. Operational programs undergo changes for a number of reasons:

- Corrections of latent errors
- Changes caused by time, such as year or tax rate
- Changes in parameters

STATE STREET BANK & TRUST CO.
DEMAND DEPOSIT APPLICATION SYSTEM
DAILY TRANSACTION MERGE PROGRAM

TABLE OF CONTENTS

Fig. 5-12*a*. Table of Contents—Sample Program. (*Courtesy*, The State Street Bank and Trust Company *and* The Diebold Group, Inc.)

- Changes in output format, or procedure caused by a change in management requirements
- Functional expansion

Because of the nature and speed of modern business, the changes which have to be made often are required with only limited notice. As a result, the physical program may be rapidly and drastically changed in a time

PROGRAMMING MANUAL

TABLE OF CONTENTS

*Items marked with an asterisk are a part of the Operating Manual

Fig. 5-12*b*. Table of Contents—Sample Program. (*Courtesy*, The Bowery Savings Bank)

period which does not allow for the proper changing of all related materials, such as documentation, listing, test data, etc. The documentation often does not reflect the latest changes, and so loses its reliability for

any further reference. The entire investment in proper documentation is lost, as soon as a change is not incorporated properly. A correction to an actual program deck which is not reflected in the program listing will automatically preclude the use of the listing as accurate in all further changes. It becomes necessary in this case to go directly to the deck, to obtain the real status of the program.

It is necessary to set up and enforce a realistic control over all required changes. An analogous situation exists in the maintenance of a blueprint file in a manufacturing company, where a separate department (Engineering Change Administration) guarantees that only the latest revisions of the prints are used. It is inconceivable to build a machine or instrument with out-of-date prints. It should be inconceivable to run a data processing system with an out-of-date program or to attempt a change to a program whose documentation is not current.

Change Procedure

A change may be initiated by a user when an output change is desired, or it may originate in the programming group when an error is found or a correction otherwise deemed desirable. The following procedure should be followed in making the change:

1. Establish a date or time when the change is to become effective. If the change is caused by an error the effective date may be immediate; if not, a date which causes the least interruption must be selected based on the length of time required to make the change and its relative importance.
2. Determine a cost for the change and a lead time, and advise the user; this will insure that all concerned are aware of the cost of changes, and may ultimately reduce the number of changes requested.
3. Determine the effect of the change on the program or programs, and ascertain the best approach to making the change. In general, there are two methods in which a change can be incorporated: a memory correction can be made (a "patch"), or the program can be recompiled or reassembled after the changes are made in symbolic notation. The decision whether the change should be in machine language or symbolic rests with the programming supervisor, and should be based on the following factors:

The number of memory corrections already made
The size of the change
The reassembly time required
The installation date of the change
The effect of the change on operating efficiency
The available memory space.

A tight set of specifications which guide the supervisor in making a change could read as follows:

Reassembly is necessary if one of the following holds true:

The last two changes were made in machine language. (i.e., there shall be no more than two memory corrections in a program at the same time)

The change affects more than one block of the program

The change exceeds twenty instructions

The change will materially affect the timing of the run.

4. Fill out the first section of a change request form. [See Figure 5-13.] This embodies the reason for the change, its purpose, and its effect, and the method to be used to implement it.

5. The assigned maintenance programmer shall then make the change, based on the outlined method.

6. Create the necessary test data to properly test the effect of the change. Incorporate this test data with the test data already established for the program, and completely test the program using *all* the test data. This will not only establish the correctness of the change itself, but also the fact that it did not adversely affect any other sections of the program.

7. Update the existing documentation to show the effect of the change. If the change is a minor correction which does not affect the program logic, the only entry required may be to enter the date and nature of the change on the revision page. (All changes should be entered on the revision page, whether the documentation was affected or not.) If the change is a major correction, or has altered the program logic, it should be reflected in all aspects of the documentation, including

The program listing

The operating manual

The programming manual

The test data

The program deck

The memory dump

The program tape

The symbolic entry card deck.

Whether the change is a memory correction or a recompilation should not affect the information that requires updating. The symbolic entry should be changed in any case, so that a future change which will require compiling will also embody all previous memory correction. [Figure 5-5, page 120, shows the form to be used to make a concurrent symbolic and machine language change, both when testing and when performing program maintenance.]

8. On the effective date of the change, replace the obsolete information with the current or new material. If the change is an emergency, effective immediately, the normal procedure would be to suspend operation of the program by removing it from the library or by placing a "hold" on it. The introduction of a new revision can then be made by merely replacing the revised materials and releasing the hold.

9. As a part of the program library control system, the change should be accepted by the librarian, after insuring that all material is properly updated. It should be recorded on the program history card, and the change request sheet should be filed by program, in revision number order.

PROGRAM CHANGE RECORD

Page____of____

Program Name_____

Change Initiated_____
By_____Date_____ Program Number_____ Change Number_____
Dept._____

Programmer_____ Effective Date_____

Description of Change:

Affected Sections:

Main Routine	
Subroutine	
Output layout	
Timing	
Halts	
Micro flow	
Memory Layout	
Tape Use	
Operator tests	
Other programs	
Other	

Previous change_____

METHOD OF CHANGE: ☐ PATCH ☐ RECOMPILE Supervisor's Initial

II.	BY	DATE	
Machine Language Working			Master Statements Adjusted
Master Statements Adjust			Compile
Listing			Micro-flow
Micro-flow			Manuals
Manuals			Test Data
Test Data			Tested
Tested - Before & After			Results Approved
Test Results Approved			Old Listing Destroyed
Other Changes			Old Deck Destroyed
Master Tape Changed			Master Tape Changed
			Librarian_____
ER 2764			Date_____

Fig. 5-13. Program Change Record. (*Courtesy*, The Bowery Savings Bank)

CHAPTER SUMMARY

A continuation of Chapter IV, this chapter illustrates methods standards for the tasks remaining in programming:

Desk Checking
Assembly
Program Test
Production Test
Systems Test and Parallel Operation
Development of Standard Techniques and Subroutines
Documentation
  Operating Manuals
  Programming Manuals

Questions for Review

1. What is the best way to prepare test data?
2. What type of test data must always be prepared?
3. What is the most important principle of documentation?
4. Why do you think it is necessary to separate the programming and the operating manuals?
5. Do you feel that "open shop" testing is better than "closed shop"? Defend your position.
6. What other items would you include as a part of the documentation of a program?
7. Should there be separate systems documentation, or do you feel that the system can be explained as a synthesis of the programming manuals?

Chapter VI

METHODS STANDARDS: OPERATION

INTRODUCTION

The operating function is in many ways a more difficult function from the point of view of management control than programming. Operation of the computer is an intermittent function because the machine controls its own operation for a good part of the time. The rest of the time, however, severe demands are placed on the operator. Machine time is costly and delays caused by humans are disproportionately expensive. The operator may cost the company $4 per hour; it is rare that the computer hour costs less than $40, and more often the cost is close to $400.

Equipment operating efficiency must be instilled in the operating staff members right from the outset. The continued operation of the computer should be of prime importance. The machine must have the *respect* of the operator, if he is to provide it with effective operation.

A programmer may spend days or weeks refining a program to save milliseconds per record. In the same installation the operator may stop the machine to step out for a coffee break, thus negating many times over the effort of the programmer. This inefficiency, and the inefficiency which occurs during set up and take down, usually has its roots in the early training of the operator. At that time, when the machine was first installed, many of the programs were not ready for operation. The programmers spent many long hours on the console, testing the programs. The manner in which these tests were operated and the inefficiencies of this critical period are often carried forward into future operating practices. This is unavoidable if rigid discipline is not established from the beginning.

A secondary factor that often causes poor operating practices is the general attitude of company management. Management's attitude, and the type of company which it represents is often reflected in the appear-

ance and operating efficiency of its computer center. (And, of course, in any other critical operating group.) Management must enforce the basic rules of cleanliness, neatness and effective operation. These rules should be written, made available to all operators, and installed as early as possible. The following sections describe some of these rules, and indicate good operating practices and formal methods standards necessary in any installation. The functions discussed include

- Housekeeping
- Machine Time Logging
- Control Functions
  - Scheduling
  - Data Coordination
  - Dispatching
  - Data Control
  - Report Distribution Control
- General Machine Operation
- Emergency Procedures
- Supply Functions
- Program Library Operation
- Tape Library Operation

GENERAL HOUSEKEEPING

Much as you can tell the quality of a chef by examining his kitchen, you can tell the quality of an operating department by a cursory examination of the machine room. The room reflects the attitude of the operators and supervisors. A neat room shows concern with appearance and sufficient interest to replace leftover paper, pick up loose cards, and straighten out desks and working surfaces. Operating standards should include rules to state the general policy in this area, as illustrated below:

1. There shall be no smoking in the machine room. The major dangers of smoking are fire, tape reading and writing problems, and its contribution to a lack of orderliness.
2. The machine room shall be kept orderly at all times, day or night. There shall be no loose papers, cards or other materials left on the floor at any time. Excess cards removed from the punch hopper must be replaced in the card storage rack. Printer forms left over at run completion must be returned to the operator in charge of forms supply. Initial sheets with line-up test lines and excess paper removed from the output are to be thrown away in the non-conservation receptacles. Excess cards, or card files to be destroyed shall be thrown in the conservation receptacles. Nothing other than punched cards may be thrown in the receptacles marked conservation. Remember: punched cards sold after our use may bring as much as $100 per ton.

3. Machine room operators shall be neatly dressed on all shifts. The computer is an important part of the Corporate image, and executives of the Corporation often visit the Center with important customers. It is important that dress shirts and ties be worn by male operators. Females shall not wear slacks or other unsuitable attire.

4. For safety and control there must always be at least two machine operators in the room when power is "on" in any unit.

5. For safety, all operators must wear tie clasps. Identification bracelets, large rings, and similar jewelry should not be worn while operating machinery.

6. Operators reporting for work under the influence of alcohol will be sent home and not paid for this absence. Repeated incidents make the offender subject to dismissal.

7. Operating personnel under medication that tends to dull reflexes will be excused from direct machine operation and reassigned to clerical work within the department during this period.

8. Cards and program decks used for operation shall be properly replaced immediately after using. The program decks must be returned to the program library; all other cards are to be kept in trays, and returned to the data coordinator.

9. The operator is fully responsible for the operation of the machine to which he is assigned. He must, as a part of this responsibility, maintain the cleanliness and orderliness of the machine and its components. When necessary, he shall clean the external parts of the machine as prescribed by the manufacturer and remove card and forms dust from the card reader, card punch and printer.

Figure 6-1 shows a machine room layout that carries out the concept of neatness.

Fig. 6-1. A Computer Installation. (*Courtesy*, Data Processing Division, International Business Machines Corporation)

MACHINE TIME RECORDING

If equipment performance is to be accurately evaluated, careful records must be kept of the exact utilization of the machine. These records often must be maintained by the operator and recording methods should be carefully spelled out.

The major objectives of careful recording of time include

- The ability to analyze equipment utilization and measure actual equipment performance against expected performance
- The ability to evaluate individual performance of operators and of the entire operating staff
- The recording of chargeable time for purposes of accounting, so that each user may be charged proportionately
- The recording of chargeable time, so that the manufacturer can be paid the exact machine rental due rather than an inaccurate and costly estimate.

Chapter VIII discusses the first three objectives in more detail. The last reason is often the most important from the economic viewpoint. If the manufacturer is being paid overtime rental, it is usually calculated on a per hour, per equipment unit basis. A manual calculation often includes a bias which, strangely enough, may cost the company a great deal of money. This bias is in part made up of operator reaction time in recording, and often made up by forgetfulness, in delayed recording. An interesting experience was uncovered by one large computer user, who when switching from a manual recording system to a machine recorder, suddenly found his overtime bill reduced by 20%!

Methods of Recording Machine Time

A number of methods are available for recording elapsed machine time, and for charging it properly to the category of utilization and the user. These include

- Manual recording of time assisted by a wall clock
- Manual recording of usage, with time stamped by a punching clock
- Manual recording of time assisted by an elapsed time recorder
- Machine recording of time assisted by the operator
- Addressable memory clock that records all information

Manual Recording, Based on a Wall Clock.—This is the most frequently used method of machine time recording, primarily because it is reasonably exact and is the most inexpensive. Rapid computation of elapsed time (the time between the recorded starting and stopping time) may be done on a 24-hour basis much in the manner of the Armed Forces. This is usually accomplished by using a 24-hour clock, available from most clock manufacturers, which gives time in increments of 24 hours, the smallest unit being .01 hour.

Recording is usually on a form which provides for the writing of the starting time, the stopping time, the job number, the user, and the category of utilization. This category may be coded for later keypunching or it may be written by the operator. Figures 6-2 and 6-3 illustrate two types of forms which are used with manual recording. The first is the more detailed log, showing various identifying information items. The second is provided free from a manufacturer of forms, and is simpler than the first. The latter shows many fewer categories of use, and may be inadequate for detailed performance evaluation.

Manual Recording, Machine Stamped Time.—Where the record will serve as the source document for time charges, or where the precision of a manual clock or reading accuracy is doubted, a punching clock may be used. Of necessity, the forms used in the clock must be dimensioned exactly as the clock requires. Some forms automatically align themselves by not only punching the time, but also by punching a locating hole. Others must be manually aligned by the operator, before the time is punched. There are many manufacturers of time stamping equipment, among which are the Cincinnati Time Recording Co., and the Simplex Co. The devices are available recording in minutes and hours, or in hours and hundreds of hours. A more elaborate device is available that records time in punched holes and uses an elapsed time computer that not only punches completion time but also elapsed time.

The main advantages of the use of such devices are the permanent, authenticated record that is provided, and the accuracy of recording, that although not guaranteed, is improved over a manual reading. Figure 6-4 shows a format usable with a time clock.

Manual Recording, Elapsed Time Recorder.—Several companies manufacture elapsed time recorders; these devices record the time when the computer is actually operating. The basic reason for using such a recorder is to arrive at a total time for which the user is to be charged; that is, elapsed productive time. Since this type of recorder only provides a record of productive time, all other categories of use must be manually recorded. Figure 6-5 shows a format of log record that can be used with either an elapsed time recorder or a time stamping clock.

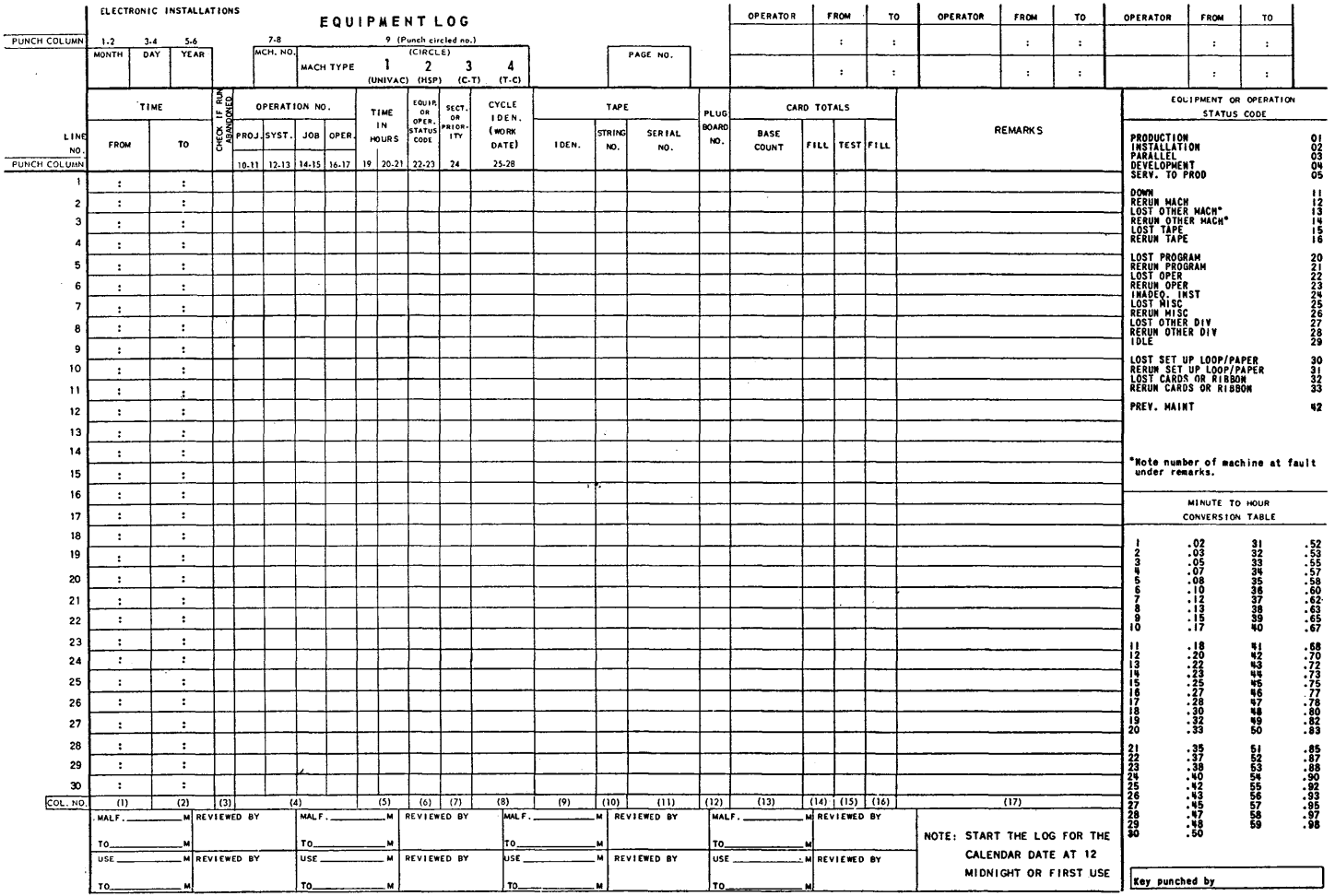


Fig. 6-2. Equipment Usage Log. (*Courtesy*, The Metropolitan Life Insurance Company)

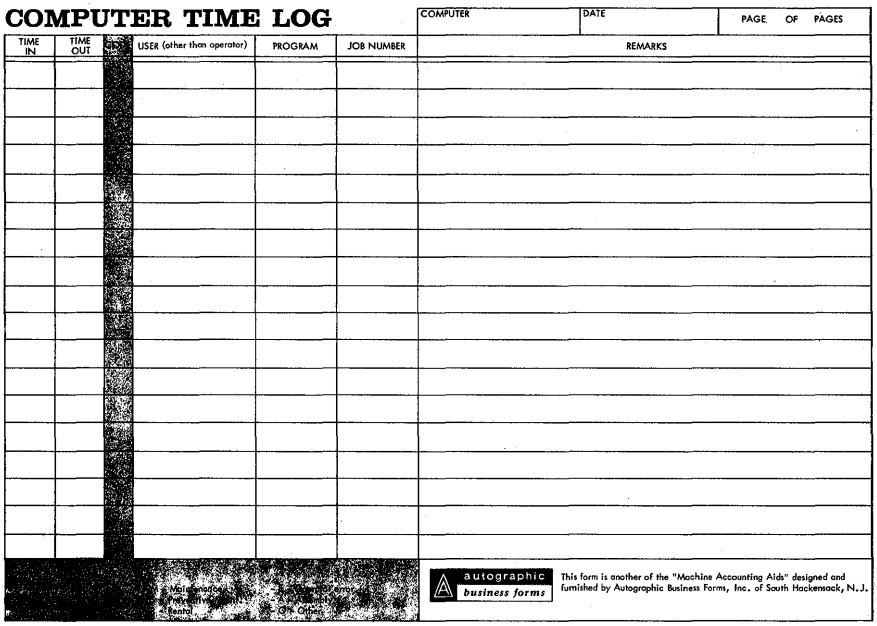


Fig. 6-3. Computer Time Log. (*Courtesy*, Autographic Business Forms, Inc.)

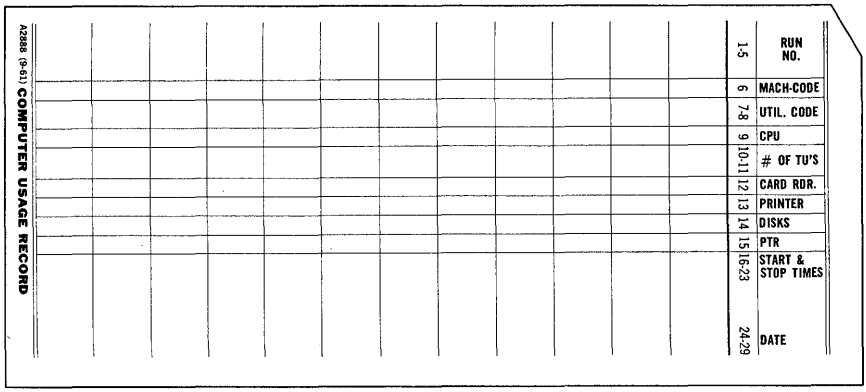


Fig. 6-4. Machine Log for Time Stamping. (*Courtesy*, General Dynamics/-Astronautics Division)

Figure 6-6 are photographs of one type of elapsed time recorders, recording time to the nearest .01 hour of usage.

Machine-Recording, Semi-Automatic.—The important discipline that must be observed in manual recording of time is that the time must be recorded as close to the instant of occurrence as possible. It is never accurate—reaction time alone is a factor in punching in and out times.

DAY NO. OF YEAR ..... INSTALLATION ..... OF .....

DATA PROCESSING MACHINE LOG

MACHINE NO. ..... – ..... DATE: ...../...../.....
MACH. DASH MO. DAY YEAR

	NUMBER OF TAPES		DAY OF WK.	STATUS	TROUBLE	TIME OF DAY		ON-LINE COMPONENTS							JOB NUMBER		NOTE #
	HIGH SPEED	LOW SPEED				HOUR	1/100 OF HR.	TAPE CONTROLLER	READERS	PRINTERS	PUNCHES	DRUMS	EXTRA MEMORIES		BASIC	DASH	

Fig. 6-5. Machine Log for Elapsed Time Recording. (*Courtesy*, Lockheed-Georgia Company, A Division of Lockheed Aircraft Corporation)

Often the operator is not near the machine when it stops; an audible halt signal installed on the machine may help but still requires almost immediate reaction from the operator. Interestingly enough, the punching of start time is almost always done on time; in order to start the job the operator must be at the console. Since stopping time is recorded less accurately, the net effect of the delay is to extend the average use time and thus to increase the amount of overtime. Many companies are therefore using machine operated recorders that indicate each starting and stopping time.

These devices vary; the most common is the direct elapsed time recorder discussed previously. This totals use time but does not show individual starting and stopping times or the amount of power "on" time, when the machine is idle or being set up. Other types of recorders operate in a variety of fashions—the Data-Timer optionally produces a printed record, a punched card, or a graphic representation. The bar chart recorder produces a graphic representation, and can handle numerous separate components of the machine, or several machines at the same time. This equipment is illustrated in Figures 6-7, 6-8 and 6-9.

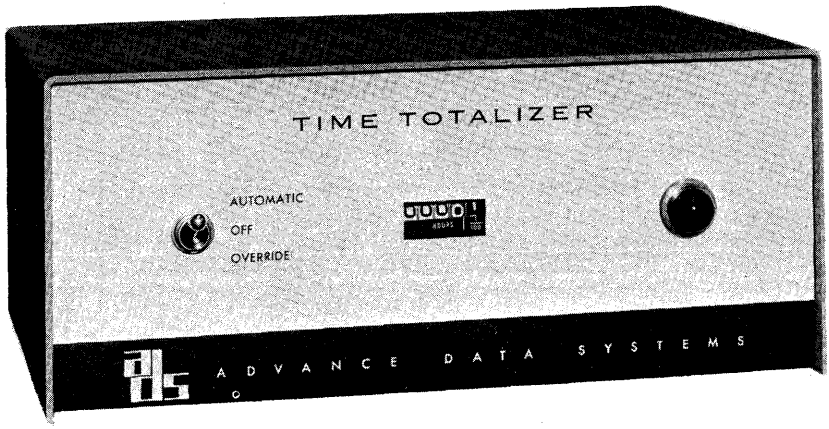
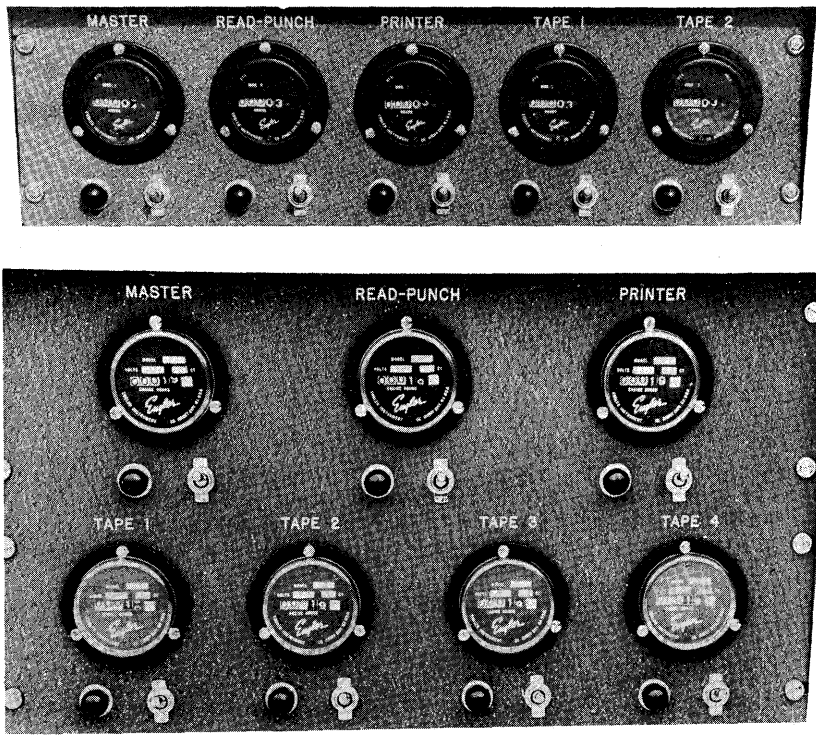


Fig. 6-6. Elapsed Time Recorders
a. Single Unit. (*Courtesy*, Advance Data Systems, Inc.)



b. Multiple Units (*Courtesy*, Engler Instrument Company)

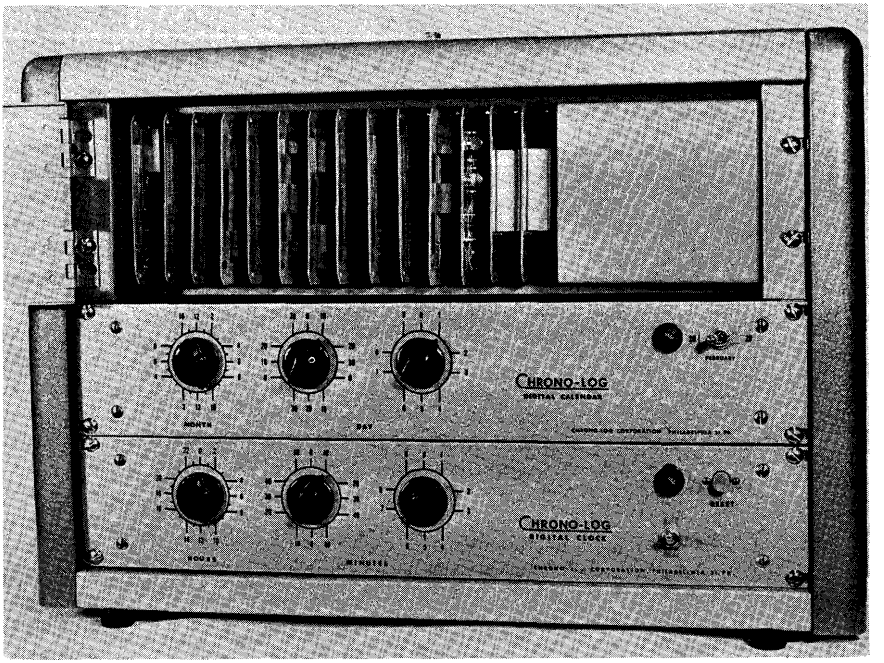


Fig. 6-7. Direct Input Digital Clock—Addressable. (*Courtesy*, Chrono-Log Corporation)

No matter what the method is, the operator must still fill in the category and the specific user. The last method of measurement, discussed below, even allows this to be done by the machine.

Addressable Memory Clock.—The most accurate device for time recording, which includes all of the information necessary, is the addressable monitor clock, built into the machine as a part of the hardware, with a separate program or subroutine to internally produce a log record. The basic requirements for such a clock are that it increments itself as time elapses and resets itself every 24 hours. Such clocks are standard and necessary equipment on all real-time computers, where elapsed time may force a specific type of interrupt. They are optional equipment on all other computers, available either from the manufacturer, or from a separate manufacturer who will make the installation.

This clocking system operates under complete program control. As such, it is capable of creating a record (on punched card or tape) of each job, each user, the elapsed time, and the amount or volume of information produced. This makes possible the detailed analysis shown in Chapter VIII, and provides for the ready reconciliation of machine time.

Unless an addressable memory clock is used, the operator is still the

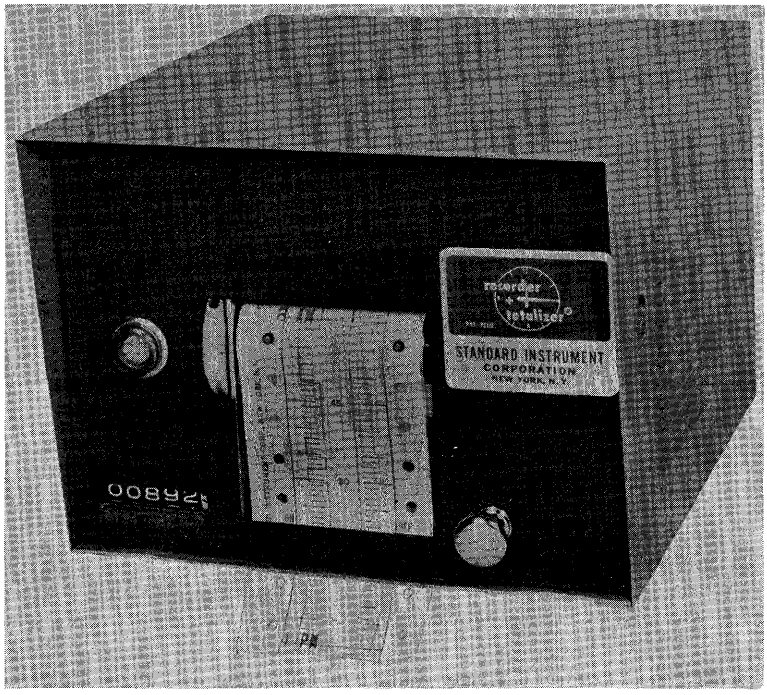


Fig. 6-8. Printing and Elapsed Time Recorder. (*Courtesy*, Standard Instrument Corporation)

pivotal point in the recording of time. He must be made aware that this is a major part of his responsibility, and that variances, calculated periodically, will be attributed to his failure in accurate logging of the machine time or its category. The standards manual should include rules which relate to this responsibility, such as the following:

The operator has full responsibility for logging of machine time at each point in time when the equipment status changes; that is, when the machine stops or is started. The information that must be recorded at the instant of occurrence is: time of occurrence, type of occurrence, category code, user and equipment used.

CONTROL FUNCTIONS

The operating department, or an affiliated operating section has five basic control functions. These five functions are:

- Scheduling: assignment of job priorities, and the establishment of a daily equipment schedule

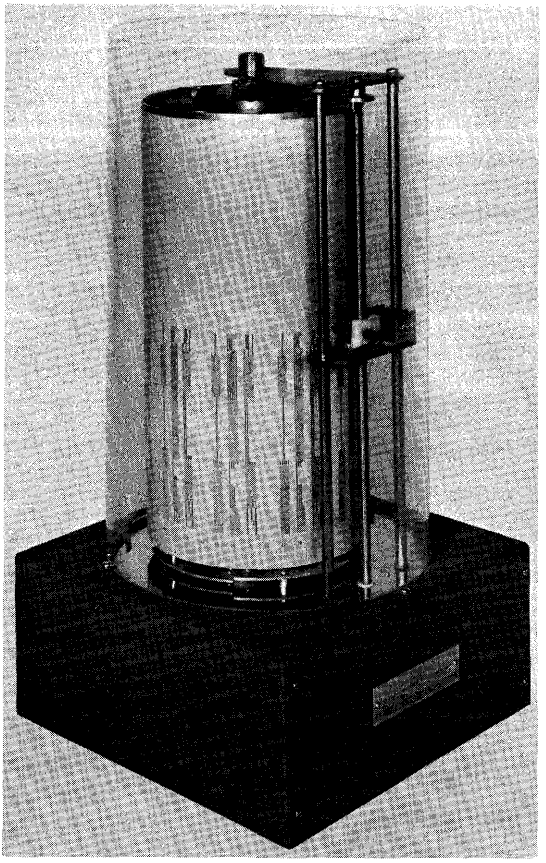


Fig. 6-9. Bar Chart Recorder. (*Courtesy*, EAI—Electronic Associates, Inc.)

- Dispatching: insuring that the jobs are performed, in the assigned sequence
- Data coordination: obtaining of all the required input and output information, and making it available to the operating group
- Data control: validation of output against predetermined totals
- Report distribution control: insuring that all reports, deleaved, decollated and bursted as required, are distributed to the competent authority.

Scheduling

A schedule for the operation of a computer is usually established weekly, although the schedule reflects each day's processing separately. The schedule must account for all activities that can be anticipated:

recurring production, special production on the basis of requests, testing, assembly, preventive maintenance, training, and demonstration.

The first step in the establishment of a schedule is review and processing of daily requests for production and testing time. The requests are submitted on special forms, two examples of which are shown as Figures 6-10 and 6-11. The first is a request for a normal production job, carrying an estimate of the amount of time which it will run. The second is a special request form, showing the request made for a program test, an assembly or a special operation, used at a commercial installation where normal production is set up on the basis of frequency alone, and no recording request card is used.

After the total number of jobs to be run has been established, the scheduling group assigns priorities, on the basis of the requested time, the urgency of the request, the user, and other factors that may be relevant. The general priority system establishes a "rush" (first priority), a normal processing operation (second priority), a test or assembly having a predetermined turnaround time, and an "immediate" priority, for jobs that must be run the instant they are received. The last is generally not scheduled; the schedule may allow buffer time for these. If no buffer is provided, it will have to be absorbed in normal processing, at the end of the processing period.

The form used for the schedule is not of great import; a typical schedule is shown in Chapter VIII in the discussion of equipment performance standards, page 232.

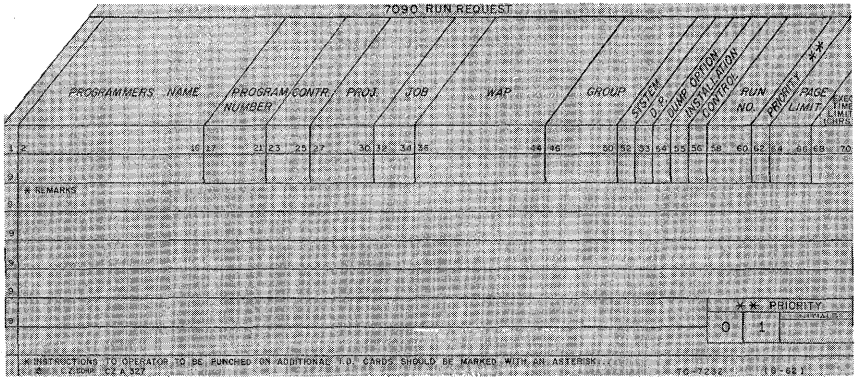


Fig. 6-10. Production Request Ticket. (*Courtesy*, General Dynamics/Astronautics Division)

EDPM REQUEST FOR: ☐ TEST ☐ ASSEMBLY ☐ SPECIAL CONTROL NO. .....

JOB NO. ..... DATE ..... PRIORITY .....

REQUESTER: ..... COORDINATOR: .....

PLANNING REMARKS:

MEMORY PRINT: ☐ YES ☐ NO

OPERATOR REMARKS: OPERATOR: .....

☐ SEE BACK

RUN TIME FOR TEST OR PRODUCTION: ESTIMATED: MIN/HRS				ACTUAL: MIN/HRS			
NOTE: ALL INPUT AND OUTPUT TAPES MUST BE SHOWN, EXCEPT FOR GEM TESTS							
	ADDR.	P/L	TAPE PRINT		ADDR.	P/L	TAPE PRINT

PLANNER'S EXPLANATION OF TEST RESULTS/COORDINATOR'S COMMENTS:

SIGNATURE:

GA FORM GD864-3

Fig. 6-11. EDPM Request. (*Courtesy*, Lockheed-Georgia Company, A Division of Lockheed Aircraft Corporation)

Dispatching

The dispatching, or expediting, function is often performed by the supervisor of the operating group. It is his responsibility to see that the schedule is observed and delays properly accounted for. A routing

slip, or ticket, is generally made out so that the operator will have the necessary instructions. The routing ticket must accompany the input and output data; Figure 6-12 shows three typical tickets.

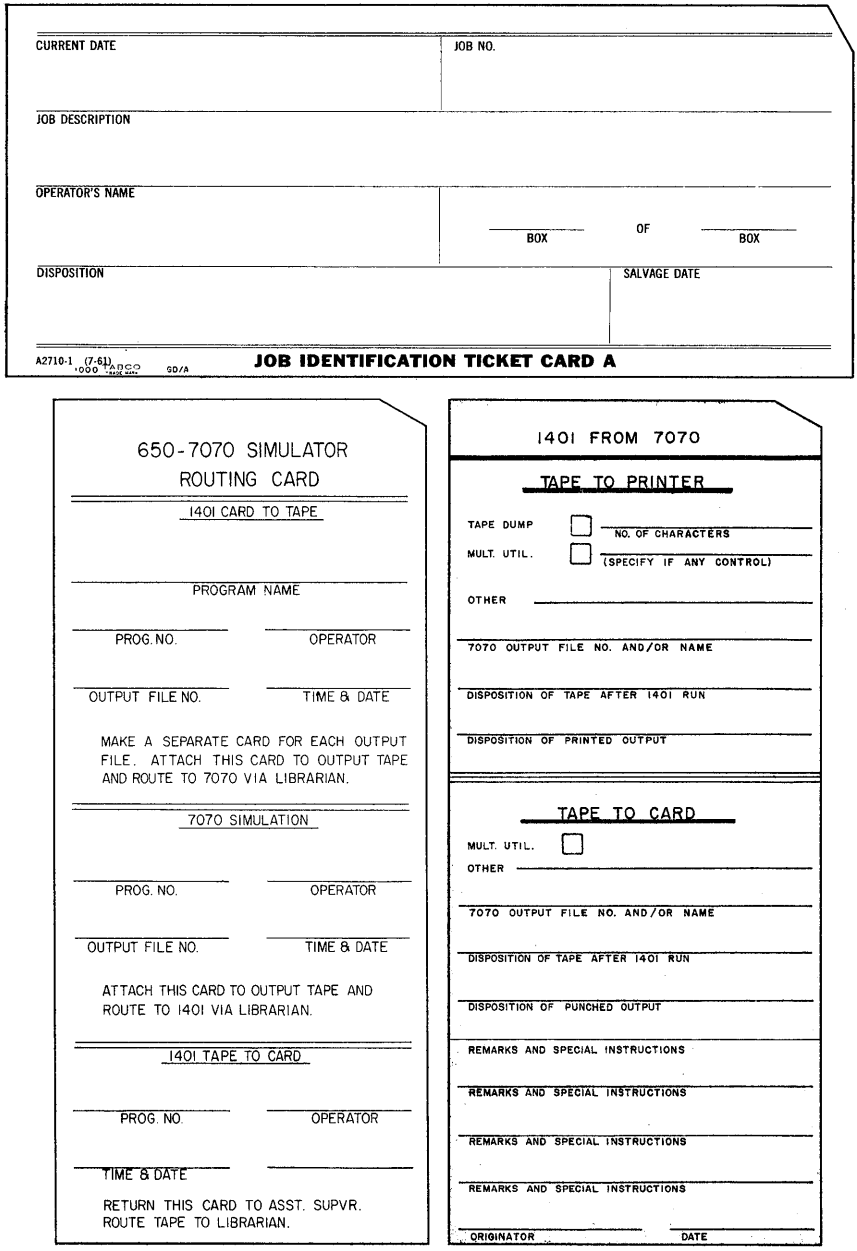


Fig. 6-12. Job Routing Tickets. (*Courtesy*, General Dynamics/Astronautics Division)

Data Coordination

The data coordination function is often combined with dispatching. The data coordinator's objective is to see that all materials required for a job are gathered in one place so that the operator can take over without delay. It is not economical to keep the machine waiting for information, while the operator is out getting tapes or forms; the data coordinator therefore has an extremely valuable function. He must obtain the following information for every job, on the basis of the documentation provided:

- Program deck or tape
- Input cards, if any
- Parameter cards required, if any
- Input tapes
- Blank cards for possible output, if any
- "Free" tapes for output and output tape labels
- Necessary stock or custom printer forms
- Any required carriage control tapes
- Job documentation
- Any other information required

This may be done according to the documentation kept in the library, or it may be done on the basis of a separate job dispatching sheet, maintained by the dispatcher or the data coordinator. This type of form is illustrated as Figure 6-13.

Data Control

In many installations computer output is separately validated. This validation is not performed item by item; it is done through the use of control totals or hash totals. This function is an integral part of the responsibility of the operating group. The output must be verified to detect machine failures and omissions of operation, data, or other vital functions. Controls are maintained on money fields by carrying group totals. On non-money fields hash totals or check sums are carried and verified to a predetermined value.

Report Distribution Control

A fundamental principle of Parkinson's Law is that work increases to meet the available time. This is especially true of computer time, and encourages in geometric proportions the production of output. A

1401 ☐
1410 ☐
7070 ☐

RUN SETUP

TEST ☐
PRODUCTION ☐

PROGRAMMER

PROGRAM NAME | PROGRAM NUMBER

CHANNEL 1						CHANNEL 2					
TAPE UNIT	PROG/FILE NUMBER	DESCRIPTION	ALT. UNIT	I/O OR S	RET. DAYS	TAPE UNIT	PROG/FILE NUMBER	DESCRIPTION	ALT. UNIT	I/O OR S	RET. DAYS

CONTROL CARDS: YES NO I.D. _____ | 1301 DISK ON ☐

7070: ALTERATION SWITCHES ON				ACCUMULATOR OVERFLOW SWITCHES ON				
1.	2.	3.	4.	1.	2.	3.	EXP.	

1401: SWITCHES ON

I/O	A	B	C	D	E	F	G	READ	PUNCH

1402:

	POCKET	DESCRIP/FORM	POCKET	DESCRIP/FORM	POCKET	DESCRIP/FORM
READ						
PUNCH						

1403: FORM_____ FIRST PRINT POS. _____ LAST PRINT POS. _____

CARRIAGE TAPE STANDARD ☐ OR (SPECIFY PUNCHES) _____

DISPOSITION
INPUT _____
OUTPUT _____

RECEIVED

SUPERVISOR _____
PROG. MAINT. _____
TAPE LIBRAR. _____

7070 CONSOLE INITIALIZATION:
0000 – 8101010004
0001 – 0100030010
0002 + 5100110002

DATE _____

A3141 (4-62)

Fig. 6-13. Run Set-Up. (*Courtesy*, General Dynamics/Astronautics Division)

recent study conducted by a large company indicated that in only three more years of continued growth of computer output the computers would be producing 30 pages of paper *per day* for each man or woman employed by the company.

As a result of this geometric expansion, one of the more important functions to be performed in an operating department is the handling of printer output. This includes margin removal, decollating, bursting, and binding, but, more important, it also includes accurate distribution of copies to all personnel indicated.

To perform this task satisfactorily within the time scheduled a separate distribution sheet is often used in the report control section for each job or procedure. Such a form is shown as Figure 6-14.

This function is becoming more and more important as printers become faster and more versatile. A 1000 line per minute printer produces 1,200 pages per hour (printed 50 lines to the page). This will produce the staggering total of 420,000 pages of information, if left to its own devices, for a two-shift operation in one month! The trend toward exception reporting will reduce the volume of information required but will correspondingly increase the importance of accurate distribution and control over confidential or secret information.

DISTRIBUTION SHEET

JOB TITLE _____

JOB NO. _____ PROCEDURE NO. _____ PROGRAM NO. _____

OUTPUT NUMBER	FROM: SECT	FROM: CHART	FROM: STEP	OUTPUT TITLE	COPIES	DESTINATION	TEL. EXT.	DELEAVE	BURST	BIND	STACK	OTHER	REMARKS

A3171 (4-62)

Fig. 6-14. Report Distribution Sheet. (*Courtesy*, General Dynamics/Astronautics Division)

GENERAL MACHINE OPERATION: NORMAL CONSOLE PROCEDURES

The Manual of Standards should contain a section describing the basic rules of normal console operation and the responsibilities of the operator at each step. A number of rules are suggested below.

Use of Documentation

1. The operator shall at all times follow the directions in the operator's manual prepared for each job. If a direction contradicts the rules of the Department or good operating practice, or is inefficient or not well thought out, the operator should bring it to the attention of his supervisor so that it can be corrected. Any operator action performed independent of the operator's manual or the Manual of Standards will be considered a severe violation of responsibility.
2. If the program reaches a programmed halt, the operator must look in the appropriate section of the operator's manual, unless
   he is *completely* conversant with the instructions printed in the manual, or
   the halt reached is a standard halt, documented in another manual, or, obviously,
   the halt indicates that end-of-job has been reached.

Program Set-Up

3. The operator shall perform all set up in the exact sequence prescribed by the operator's manual. In this manner advantage can be taken of the maximum overlap.
4. Tape set-up. The operator shall mount the required input tapes on the designated drives and free tapes on the drives designated for output.
5. Tape cases shall remain with the reels with which they came. The reel removed from the drive shall be replaced in its own case and then placed on a special rack adjacent to each tape unit. Cases must remain with the reel so that damage can be easily traced.
6. All tapes when replaced in the case shall contain a "grommet" or other device to prevent the tape from unwinding.
7. Immediately upon removing output tapes, the write inhibit or file protect ring shall be removed from the tape. The operator shall insure that there is no ring in any input tapes.
8. Printer set-up. When using a stock form, the operator must ascertain that the width of the form is sufficient to prevent printing on the back-up platen.
9. When using a custom form, or a stock-imprint form, the operator shall use the programmed line-up routine to insure that printer-to-form alignment is within 1/32" of perfect registration. Reports produced outside this registration tolerance will be rejected, and must be rerun. The required carriage tape must be checked before it is mounted.

10. Card reader set-up. The operator shall place all designated card files, in the appropriate sequence, in the card reader; the operator shall insure that all parameter cards are correct, that a date card is included if required, and that the entire card file is free from imperfections, which may cause a card jam. The operator shall joggle the cards sufficiently to insure that rubber bands or other extraneous materials have been removed from the file.

11. Card punch set-up. The operator shall insure that a sufficient supply of unpunched cards are in the card punch, if used. This supply must conform to the proper card design specified. If none is specified, the operator shall use blank cards—Form # *XXX*.

12. Console set-up. The console must be set up in exact accordance with the instructions provided in the operator's manual. The operator shall check each item in sequence. . . .

13. If an engineering console is provided, or a separate set of switches are available which can change the mode of operation, the operator is responsible for insuring that these have not been set improperly. This has to be done only after a shift change, or after the maintenance engineers have turned over the machine

Normal Operation

14. During normal operation the operator must watch the processing to detect malfunctions or unusual machine actions. The operator must replenish the supply of input and output cards without stopping the machine, if possible, and remove and replace all cards the machine has read or punched.

15. The operator under no condition has the authority to alter memory of an operating program. The operator may not alter any program, program deck or program tape without the explicit approval of both the operating and the programming manager. Under no condition should an operator run any program other than one authorized for operation on the current schedule.

16. If a machine failure, data error, program error or operator error occurs, the operator should follow the instructions outlined under emergency procedures. Under no condition is the operator allowed to rerun without authorization or to use any self-constructed or utility program in an attempt to correct the file.

17. If an operator is interested in and qualified to become a programmer, he should apply for a transfer to the programming department. He may not under any condition write programs and operate them during off-hours.

Take Down Procedures

18. The operator shall remove all tapes after the program has completed its processing. Intermediate tapes may be removed after the program has completed its rewinding.

19. File protect or write inhibit rings shall be removed from the tapes immediately upon their removal.

20. Upon the removal of a completed reel of output, the operator shall immediately affix a self-adhesive permanent external label. This label identifies the reel number, the reel sequence, file number date, drive (for error tracing) and the operator's initials. [Such a label is illustrated as Figure 6-15; it can be

obtained in continuous form and prepared using a separate label writing program.]

21. All cards shall be removed from the stackers and hoppers and replaced in the appropriate location. The inputs and outputs of the job must be returned to the data coordinator for subsequent processing or distribution. The program and its documentation is returned at the same time.

22. All recording of time shall be done as previously specified.

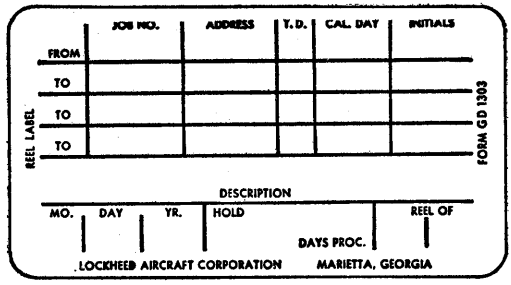


Fig. 6-15. External Tape Label. (Continuous Form/Pressure Sensitive). (*Courtesy*, Lockheed-Georgia Company, A Division of Lockheed Aircraft Corporation)

GENERAL MACHINE OPERATION: EXCEPTION AND EMERGENCY CONSOLE PROCEDURES

A specific section of the Standards Manual should be devoted to the writing of exception and emergency procedures. *Exception* procedures refer to the occurrence of an unexpected machine-caused condition: a program error, a machine failure, an operator error, or a data error. *Emergency* procedures refer to the steps to be followed in case of a real emergency: flooding, fire, electrocution, attack, and the like. The latter are rarely specified by most installations; this is unfortunate, since advance planning may save thousands of dollars in such an emergency.

Exception Procedures

1. If a programmed halt occurs, the operator shall immediately note the halt number and the status of files, cards, tapes and the like. The operator shall then look up the halt by number, to determine the cause and action to be taken. In the event the halt is "endless"—without corrective possibility, the operator shall notify the supervisor of the occurrence immediately, and proceed to the next program.

2. If a machine error occurs, and the machine stops, the operator shall immediately record the occurrence on the log. He must then notify his supervisor and the maintenance engineer. By reviewing the documentation he determines if the program contains a "restart" procedure. If so he must follow the restart instructions shown. If no restart procedure is available the operator

should start the run over from the beginning. A second occurrence shall be cause for terminating the run, and for turning the machine over to the maintenance engineer for unscheduled maintenance.

3. Whenever an exception condition occurs the operator is responsible for noting all existing conditions on the "console condition" page. The same form must be used for all exception console conditions, including the occurrence of a program error, a data validity error, or an operator console error. (This page is often designed as an image of the console enabling the operator to record the information rapidly and accurately. Two such forms are illustrated; Figure 6-16 shows the image of the console; Figure 6-17 shows information required; since the console is much more complex it requires only the entering of the contents of certain registers.)

4. In the event of a program error the operator shall notify his supervisor and the programmer responsible for the program.

5. In the event of a data error the operator shall notify his supervisor.

6. In the event of an operator error, the operator shall notify his supervisor; if the error has destroyed pertinent information, the supervisor shall also notify the programmer responsible, to assist in recapturing the required information.

Emergency Procedures

7. The automatic fire and smoke alarm systems will signal in the event of fire in the computer room. The operator *must* immediately turn the Master Power Switch Off. If time permits the operator should remove all tape files, and store these and the current program deck and documentation in the fire proof tape vault or other designated storage. No further protective measures need be taken.

8. In the event of a malfunction in the electrical system or an electrical storm which threatens to back-circuit the computer system, the operator should immediately turn the Master Power Switch Off and notify the maintenance engineers and his supervisor of his action.

9. If an operator comes into contact with an exposed electrical lead and is subjected to electrical shock, the other operator should immediately turn off power on the unit or the entire system, whichever is faster. He should then remove the stricken operator from the immediate contact area and administer first aid in accordance with the First Aid Manual.

10. In the event of flooding, or the potential of water damage, the operator shall first turn off Normal Power, if time permits. If not, the immediate Master Power Switch should be used to prevent further systems damage. All information which should be protected should be moved to high ground, wherever possible.

11. All files which should be protected at all costs shall have a "red" external label. These files, including master program tapes and the like, shall always be stored in the fireproof/waterproof section of the tape library. If any emergency occurs with a "red" label file on the system, the operator shall attempt to return this file to the library, if possible.

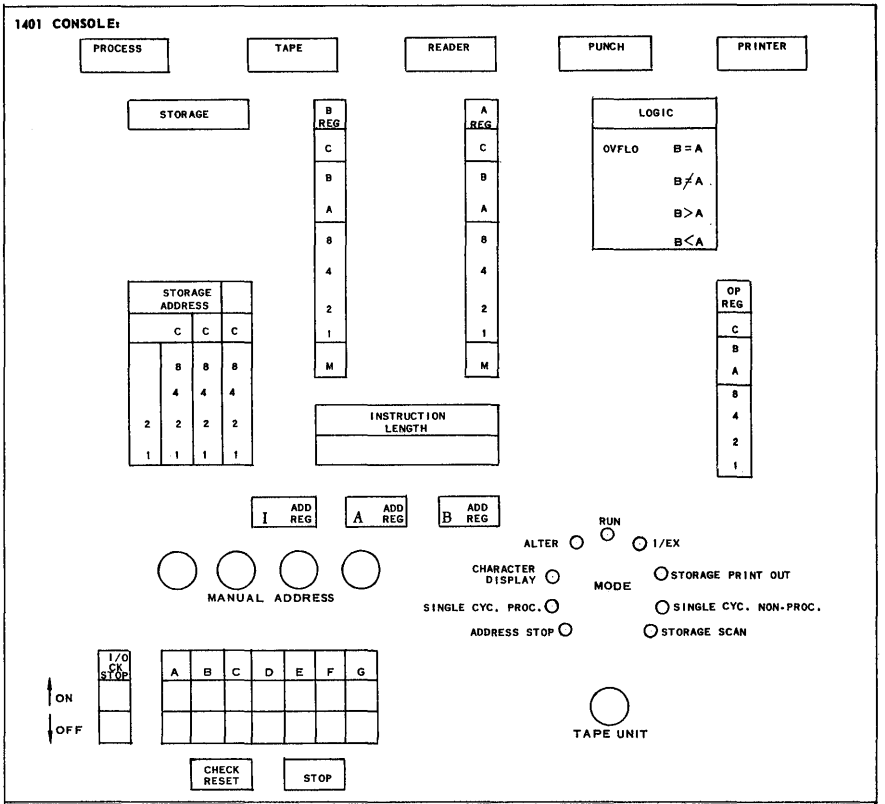
LOCKHEED-GEORGIA COMPANY
A DIVISION OF LOCKHEED AIRCRAFT CORPORATION

TIME .....

DATE .....-.....-.....

JOB NO. .....

CONSOLE SCHEMATIC – 1401



FORM GD1433A

Fig. 6-16. Console Schematic. (*Courtesy*, Lockheed-Georgia Company, A Division of Lockheed Aircraft Corporation)

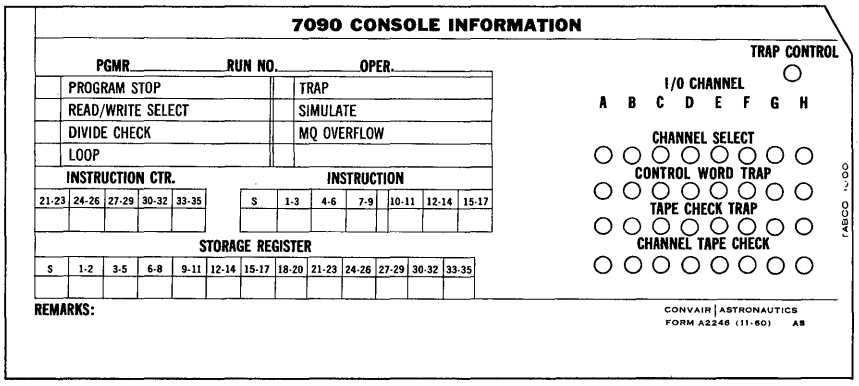


Fig. 6-17. Console Condition Recording. (*Courtesy*, General Dynamics/Astronautics Division)

SUPPLY FUNCTIONS

A minor but important function of the operating staff is the maintenance of the physical and/or perpetual inventory of the supply of forms, cards, tapes and all other information required by the computer center. If inventory records are maintained by a separate purchasing unit the operator must fill out a withdrawal slip for additional supplies. If this function is not handled by a separate group, the data coordinator, or a designated operator usually controls supplies.

This control function is important. It is obviously uneconomical to run out of forms before a reorder is given. Rush orders are more expensive, and operation without a specific form can be quite a problem. On the other hand, the cost of an oversupply of forms can run into thousands of dollars and a reduction of excess results in an increased cash flow. Also, form redesign and change is very common in the data processing function and it rarely pays to order five years supply of a 10-part form. Redesign is often not attempted only because present inventory is too large; this often prevents the use of cheaper or better supplies.

Punched card inventories incur similar problems and the cost of oversupply also includes a significant space charge. Some companies use over 100,000 cards per day; the storage space necessary to keep a month's supply on hand is hard to picture. Luckily, the cost of running out of a specific form is not as high; either a blank or other card format can be used or blank cards can be overprinted with a reproducing master.

In any event, someone should be delegated to maintain a detailed record of supply status. His function will include keeping track of orders, current usage, available space, economic lot size and minimum order quantities. He must also insure that excess forms are returned to

the supply room, and that the correct quantities are removed by the data coordinator or his representative. Pilferage is less of a problem with data processing supplies than with other items. Nonetheless, some cotnrols should be instituted to prevent waste and other shrinkage.

PROGRAM LIBRARY ORGANIZATION

The librarian generally has two functions: the maintenance of the program library and the tape library. As a program librarian the main functions which he must perform are

- Program record retention
- Release of programs to operators
- Maintenance of revisions

Simple rules for this task are listed below:

1. Upon receipt of a new program, the librarian must fill out a program record index card. [See Figure 6-18*a*.] The librarian should know that all information represented on the transmittal check list is present and has the correct revision number.
2. If a program is operational, the librarian shall release it to operations upon request of the data coordinator. The only material which can be so released is the operator's manual, and the program deck or tape. A sign-out sheet will be kept daily. [See Figure 6-18*b*].

PROGRAM NAME				NO.
MATERIALS INCLUDED: ☐ DECK ☐ SYMBOLIC DECK ☐ TEST DATA ☐ PROG. MANUAL ☐ OPER. MANUAL ☐ LISTING ☐ MEMORY DUMP ☐ PROGRAM TAPE ☐ OTHER _____				
REVISION RECORD				
NO.	DATE	LIBRARIAN	CHANGES MADE BY	AFFECTED SECTIONS
0				☐ OVER

Fig. 6-18*a*. Program Record Card.

DATE _____	SHIFT _____	DAILY PROGRAM SIGN OUT RECORD				LIBRARIAN _____						
PROGRAM NO.	PROGRAM NAME	SIGNED OUT BY	O/P	RETURN	MATERIALS INCLUDED							
					DECK	O/M	P/M	LIST	T/D	SYM	OTHER	

Fig. 6-18*b*. Sign Out Sheet.

3. Upon request by the responsible maintenance programmer, the librarian may release any or all documentation to the requester. This information will also be recorded on the daily sign-out sheet.

4. Upon being notified that a program is to be changed, the librarian shall place a stop order on the program. All requests for the program will now have to be approved by the responsible programmer before the program will be released to operations.

5. After a program change has been made, the librarian shall be supplied with a program change notice. [See Chapter V.] Upon receipt of the notice, the librarian shall review all of the accompanying materials, to insure that the change has been properly made to all elements of the program. If the change is proper, the librarian shall record the revision number and other appropriate information on the program record card. The preceding revision will now be destroyed by the librarian.

6. In the event of fire, or other occurrences which may damage the program library, the librarian shall transport all materials to the fireproof vault provided for such an emergency.

7. The program library shall be maintained in strict sequence, by application, frequency and program number.

8. When a program has been obsoleted or replaced by another system, the responsible maintenance programmer should notify the librarian, indicating that the program and all accompanying materials are to be destroyed or transferred to "dead" storage.

9. If a specific program has not been used for over two years, and no indication of obsolescence is provided, the librarian may request that the responsible programmer indicate what disposition is to be made of the materials in the library.

10. At the request of the maintenance engineer, the librarian will accept responsibility for the storage and safety of diagnostic programs. These programs may not be released except to an authorized maintenance engineer or his representative.

11. The program and tape library shall be kept locked, and keys issued only to authorized personnel.

TAPE LIBRARY ORGANIZATION

The duties of the tape librarian are quite similar in respect to the maintenance of records and the release of information to the operating department. Specifically, the tape librarian must keep records on the history of each tape and information about the files and their retention cycles, and he must protect the files and issue them only to authorized personnel at the proper time. The librarian shall maintain the records in accordance with the following rules:

1. All tapes must carry a pressure-sensitive external label to identify their contents. [See Figure 6-15.]
2. The librarian will prepare a tape record card for each tape as it is added to the tape library. The tape record card [see Figure 6-19] shall show:
   A history of the "stripping" or removing of lengths of tape from the front to reduce the occurrence of errors

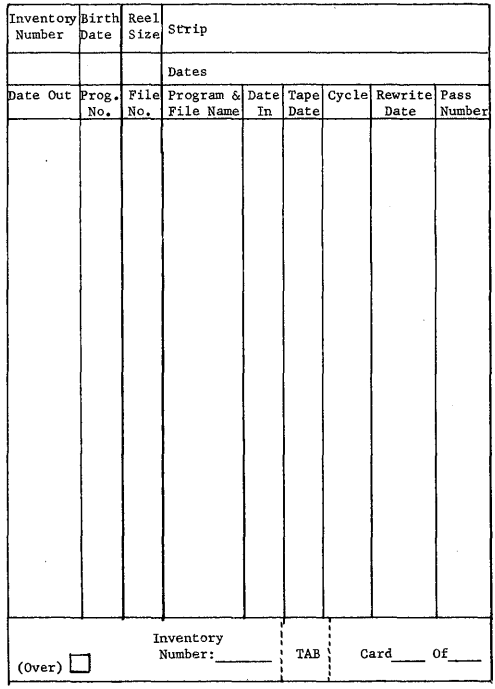


Fig. 6-19. Tape Library Record Card. (*Courtesy*, The Bowery Savings Bank)

A sign-out record to indicate what information is currently available on the tape

The assigned tape inventory number

3. The tape record card shall be kept in tape inventory sequence; this sequence shall be assigned by the librarian as follows:

Nos. 000 to 199	Application 1
Nos. 200 to 399	Application 2
Nos. 400 to 599	Application 3
Nos. 600 to 799	Application 4
Nos. 800 to 899	Maintenance Engineering
Nos. 900 to 999	Programming

4. As the tape retention cycle expires, the tape becomes available for use. At this time the librarian shall remove the external label and replace it with a new "available" label. The tape record card shall be tagged with a green index tab, to indicate that the reel is available for future use.

5. If a reel is to be saved for a special purpose or retained indefinitely the requester shall fill out a tape file save request. [See Figure 6-20*a*.] This request is attached to the tape file record card and a red index tab attached to the card to signal unavailability.

6. If the tape develops errors in the first 100 feet (this is the most likely place to experience excessive wear, because of magnetic labeling procedures which double the wear at the front) the librarian shall strip a length of tape when the tape again becomes available for use. To indicate that the tape is to be stripped the librarian must attach a blue index tab to the card. When the librarian strips the tape he must also replace the beginning marker. He then subtracts the amount stripped from the current length and enters the new length on the card. The reel is then to be marked as a "short" reel, and used only if a specific reel length is requested.

7. Tapes saved by specific programmers or maintenance engineers are to be kept in a separate part of the tape library. Each person entitled to request tapes shall have a separate tape assignment card. [See Figure 6-20*a*.] This record

SAVE TAPE REQUEST

PROGRAMMER	LOCATION	REEL NO.	UNIT
RUN NO.	**DATE**	**LABEL**	

SPECIAL INSTRUCTIONS

ET/AL	BINARY/BCD	HI/LO DENSITY

A2122 (REV. 2-61) AS

Fig. 6-20*a*. Tape File Save Request. (*Courtesy*, General Dynamics/Astronautics Division)

must be reviewed periodically by the individual to determine the tapes to be released.

8. Tapes will be stored in sequence by reel inventory number.

Storage of tapes by reel inventory number is most common, but there are installations where cycling of tapes is done in such a way that they may be stored by cycle or application. Other installations effectively use a tape reel or tape label color coding scheme providing ready identification of the files as being related either to a specific application or to a specific cycle (i.e., daily, weekly, monthly).

Cycling of tapes creates some problems in inventory control, since daily tapes would get a great deal more wear than monthly tapes. As a result, the cycling process is changed often so that tapes are reversed in their use.

TAPE ASSIGNMENT CARD

NAME		RUN NO.	DATE
UNIT	**MODEL**	**ASSIGNMENT**	

A2123 (REV. 6-61)

Fig. 6-20*b*. Tape Assignment Card. (*Courtesy*, General Dynamics/Astronautics Division)

9. The librarian is responsible for retaining an approximate idea of the number of passes to which each reel has been exposed. This may be done by having a magnetic trailer or label record on which a pass counter is recorded, or the librarian can update the tape record card as the tape is signed out and returned. After the pass count exceeds a certain predetermined figure (somewhere between 1000 and 2000 for mylar tapes), the tape, when it becomes available, shall be sent to the manufacturer for reconditioning. In installations that store tape in reel inventory number sequence, an application cross-index record is often maintained. The librarian keeps a separate record for each application or each program within an application. If a specific file request comes in, the librarian can refer to the Tape File Reference Log [Figure 6-21] to determine the reel inventory number.

PAGE ______ OF ______

TAPE FILE REFERENCE LOG

PROGRAM NAME ______________________ PROGRAM NO. ______

CREATION DATE			1401 PROGRAM NO. USED TO CREATE THIS TAPE FILE	ACTUAL REEL TAG NUMBERS		ACTIVITY DATE		
MO.	DAY	YR.		FROM REEL NOS.	TO REEL NOS.	MO.	DAY	YR.

A3170 (4-62)

Fig. 6-21. Tape File Reference Log. (*Courtesy*, General Dynamics/Astronautics Division)

10. Tapes should not be released or taken to any area except the computer center. All tapes must be transported in standard carriers and handled with extreme care. Tape testing shall be performed, if required, using the computer racks in a location with extremely careful temperature and humidity controls. The area selected for tape storage should be carefully tested for magnetic influences from surrounding equipment such as burglar alarms, elevators, and the like. The area should be fireproof and waterproof, with an hermetically sealed door with a snap-lock.

11. There shall be no smoking in the tape library. Dust shall be kept to an absolute minimum. The librarian and all others entering the tape library must not wear clothing made of angora or similar shedding materials.

12. In the event of fire or other emergency, the librarian shall lock all fireproof sections of the library and all materials shall be returned, if possible, to the proper section of the library.

CHAPTER SUMMARY

Methods standards for the operating department are at least as important as the standards established for systems analysis and programming. The rules and procedures should cover all phases of operations, from computer set-up to forms control and supplies.

The chapter has discussed some of the standards required. In addition, the chapter has illustrated methods of machine time logging, good housekeeping practices, the necessary control functions and general and emergency machine operating procedures. Each installation must, of course, develop its own procedures; the above have been shown to illustrate the kind which could be developed and installed.

Detailed standards should also be established for the operation of the program library and the tape library; these are vital functions which insure that operation continues properly, using the proper programs and the proper tape files. Management and its attitude are a vital part of enforcement of good operating standards, as much as in every other area.

Questions for Review:

1. Develop a flowchart, or a process chart, of the functions which the computer department performs.
2. Indicate the effect the following will have on good operating practices:

   computer room layout
   distance between operations and supplies
   programmer testing on the machine
   slack management attitudes
   no data coordination function
   no daily computer room schedule

3. Develop a set of records for the maintenance of a tape library.
4. Develop a detailed procedure for data control for a payroll application.
5. What are the advantages of a monitor-controlled operation? What elements of operations will then not be required?
6. What type of computer log is best suited to your requirements?
7. Should operators be able to program? Give the advantages and disadvantages of either possibility, and indicate your decision.
8. Develop a tape library procedure for filing tapes by application, within cycle.

Chapter VII

INSTALLATION AND ENFORCEMENT OF METHODS STANDARDS

INTRODUCTION

The last five chapters have outlined detailed standards for systems analysis, programming and computer operations. This chapter deals with the installation of standards and the methods needed to enforce and maintain a successful standards program.

The benefits of standards can be readily recognized; it is more difficult to *realize* these benefits, especially if management does not enthusiastically support the standardization effort. This is indeed most critical; lack of management support will destroy the best intentions.

The installation of methods standards often causes temporary problems while the entire staff is becoming aware of their benefits. If initial enforcement is not positive the entire program will be jeopardized. This kind of program should be attempted only if the following principles are understood:

- The entire program must have positively expressed support from *top* management.
- The installation must be comprehensive and put in effect in all parts of the department at the same time. Piecemeal installation is rarely effective.
- Enforcement must be positive: discipline loses its effect if it is administered in a half-hearted manner.
- The cooperation of the staff must be enlisted both in development and in installation.

INITIATING A STANDARDS PROGRAM

Step 1—Forming a Team

First, one or more staff members are given standards responsibility. This is the standards "team," whose full-time assignment will be the installation and initial enforcement of the procedures.

The team should consist of:

- One or two men (in a "small" installation) whose background includes at least one year in operation and one year in programming or systems analysis.
- Two men minimum, in a "medium" installation; one of these men should come from operations, with a minimum of two years of operating experience, at least one year of which is in a supervisory capacity; the other member of the team should come out of programming or systems analysis. In either case, the second member must have at least two years of programming experience and six months to one year of systems analysis or design.
- Three men minimum, in a "large" installation; one person from programming, one from operations and one from systems analysis. Each of these men should have a minimum of two years experience in their respective fields, with at least nine months of supervision in the area which they represent.

In addition to the team members, the data processing manager should act as the "ex officio" chairman of the team. If possible an outside consultant, or someone else with a broad standards background, should be enlisted as an advisory member of the team to act in a reviewing and advisory capacity, on a part-time basis.

Since this team will determine the data processing procedures and disciplines to be installed, it should contain the best qualified personnel. This may be difficult to accomplish because of the pressure of day-to-day business but there is little choice if a good standards program is desired. Furthermore, the team should have no other responsibilities or duties.

Step 2—Announcement to The Staff

The head of the data processing department must announce the establishment of the standards program, its objectives, and its benefits. This may be issuance of a simple memorandum or the holding of a group

meeting at which free discussion is encouraged. An announcement memorandum might say the following:

To: All members of the Data Processing Department
From: The Department Manager
Subject: The development of a manual of standards and procedures

In the continuing interest of improving our daily performance, and to strengthen the department, it has been decided to initiate a development program with the following objectives:

a. To develop a manual of standards and procedures which embodies all current standard methods and expands these wherever required.

b. To create new methods and procedures in systems analysis, programming, and operations, for more efficient and more economical performance in these areas.

Messrs. *X, Y* and *Z* have been assigned as a standards team, reporting directly to the Data Processing Manager. The Executive Committee of the Corporation has expressed a great deal of interest in this program, and we will be calling on you for your assistance in the near future. In the meantime, please direct all your questions and suggestions to me, so that they can be given proper recognition.

A group meeting for all personnel of the department is a powerful communications tool. In this event, the following steps should be taken to insure its success.

1. Invite all staff members by memorandum indicating the purpose of the meeting and that attendance is mandatory.

2. Develop a detailed agenda for the meeting, which should include the following:

- Introduction by the data processing manager
- A few words from a senior executive of the firm
- The introduction of the standards team
- The presentation of a brief list of items to be included in the standards manual
- Discussion period

3. Carefully organize the content of the material to be presented. The corporate executive should express an indication of top management's interest in the project; the data processing manager should place his personal prestige behind the project, and the standards team should request the cooperation and suggestions of all those present.

Still another method of announcing a standards program and soliciting suggestions is a questionnaire directed to all personnel. If the members of the standards team enjoy the respect of the remainder of the department, they can issue the questionnaire after project announce-

ment. In issuing such a questionnaire, the following should be stated clearly:

- What standards are
- Why they are important to the data processing program
- That management is fully behind the project
- That positive suggestions and constructive criticism will be welcomed
- That suggestions may be made anonymously

The explanation should be concise and should use examples to illustrate the kind of ideas being sought.

Typical questions are given below.

1. In what ways do you think you could best improve your performance?
2. Which aspects of your job cause you the greatest concern?
3. How do you define your responsibilities?
4. If you were to establish a rigid procedure for the performance of your job what would it be?
5. If you had to teach someone else your job, what would you tell him to do?
6. What part of your job do you feel could be made uniform?
7. Which parts do you feel could never be made uniform?
8. Outside of your own job, where do you see the greatest need for better procedures?
9. Where do you see the greatest need for more standardization?
10. What sections would you add to present program documentation?
11. What information would you add to the systems definition?
12. How do you think block diagramming could be standardized?
13. Do you think we should use more standard programs or program segments? If so, which type and what would be their function?

The above questions vary from the general to the specific. This stimulates staff thinking. Another benefit of the questionnaire approach is the number of unsolicited and unexpected comments that will be made. These suggestions should be taken at face value and, if possible, used.

Step 3—Development of An Approved Table of Contents

The standards team is now ready to go to work. The first action is to develop a draft of the Table of Contents of the Manual of Standards. This draft should be developed in as much detail as practicable. The sources of information should include:

- Current standards
- Suggested areas from the questionnaire answers or group discussion

- Standards in use by other organizations
- Standards recommended by the manufacturer
- Trade publications or other literature
- Ideas from management and supervision

It is now necessary to obtain approval from the group supervisors for each of the items directly affecting their operation. Accordingly, the Table of Contents should be drawn up with a brief explanation of the type of standards proposed in each functional area. After this a meeting should be held to obtain the suggestions and ultimate approval of the supervisory staff.

This approach has several advantages. The line supervisors are the most important to the success of the program and are directly responsible for explaining it to their staff. The line supervisors must therefore be completely sold on the program, its objectives and its results.

The line supervisors, however, can be extremely defensive about their operation, and about the necessity for rigid standards. They should not be able to object to the Table of Contents, however, since everyone recognizes the need for some regulation in each of the areas outlined. Obtaining their approval and their suggestions for the Table of Contents has the effect of obtaining their support for the entire program, regardless of the ultimate contents of the manual. Once they have approved the Table of Contents they can hardly object too strenuously to the actual developed material.

After the necessary approval for the Table of Contents has been obtained, it should be distributed to all concerned staff members with a further request for suggestions. A typical Table of Contents is illustrated as Figure 7-1.

Step 4—Development of the Contents

The major task facing the installation team is development of the contents of the manual. The success or failure of the entire program depends largely on the quality of standards developed. If they are too weak they will be ineffective. If they are too strong they will be rejected as impractical after some use.

In addition to the development of the rules which will make up the contents, the team must also develop

- The format of the manual
- The tone and phraseology
- The enforcement procedures
- The method of manual review and maintenance

Format.—The format of the manual contributes a great deal to its acceptance. As in all publications and documentation, the manual should contain

- A title page
- A revision page
- A table of contents

and it should be bound in a cover which reflects the interest of management. The manual should be appropriately sectioned, with major chapters devoted to the major functions:

- Programming Standards
- Systems Analysis Standards
- Operating Standards
- Personnel Standards
- Performance Standards

Within each chapter or major section, the detailed section should be lettered or numbered in sequence, with page numbers starting with 1 for each new section. This will enable much more flexible maintenance and expansion of the topics in the manual.

Tone and Phraseology.—The tone adopted for the manual should be consistent throughout. The most successful manuals usually are written in a slightly imperative style, i.e., a tone which prescribes the actions to be taken or the methods to be used. As an example, ". . . all block diagrams shall be drawn using $8\frac{1}{2}'' \times 11''$ paper, unruled, white stock issued under form no. . . ." The phrases used should be clear and concise and avoid the use of difficult words where simple substitutes are available. One important tool, the illustration, should not be used too sparingly, and every opportunity to illustrate the correct procedure should be used. This will have the effect of making the manual bulky; this is far preferable to a set of rules whose meaning is not too well understood.

Step 5—Management Review. The Installation "Committee"

After the draft of the standards manual has been prepared, a meeting should be set up with the line supervisors and the department manager. The purposes of the meeting are to

- Review the manual of standards
- Obtain the necessary approvals
- Establish the method of installation
- Establish the installation schedule
- Elicit further suggestions

PROPOSED TABLE OF CONTENTS FOR THE STANDARDS MANUAL

Fig. 7-1. Standards Manual—Table of Contents

H. Standards for Program Documentation

Operating Manual

Programming Manual

I. Documentation Maintenance

III. Standards for Operation

A. Tape and Program Library Organization

B. Console Procedures

Normal

Emergency

C. Scheduling

D. Record Keeping

IV. Standards for Performance Evaluation

A. Factors to be rated

B. Formulas to be used

C. Planning and Progress Reporting

V. Personnel Standards

A. Qualifications Required

B. Training Standards

C. Personnel Selection Standards

Fig. 7-1. (cont.)

In order to reduce the total time required for management review, it is a good idea to provide the supervisors with a draft copy well in advance of the meeting, so that they will be able to become familiar with the contents. It is only necessary to obtain approvals from each manager for the section which is his own responsibility; it is not necessary to give each supervisor the opportunity to comment on the entire manual.

Since the supervisors and the manager have already approved the Table of Contents, they should be fairly familiar with the intent of each section. The review meeting should not last too long, especially if each supervisor has prepared his comments in advance. The entire group should consider any changes a supervisor suggests; if the standards team agrees with the suggestion it should be incorporated; if the team does not agree, it should give its reasons, and the data processing manager will make the final decision.

After approval of the manual has been obtained, an installation schedule should be developed that takes into account

- Time required to produce the manual
- Effort required to change existing procedures
- Present and projected departmental workload

The installation mechanism should be established, possibly with the aid of a "steering committee" of the line supervisors to assist in resolving any problems.

Step 6—Develop The Final Draft of The Manual

The final draft of the manual will be the first edition to be published. It should therefore be printed in as many copies as are required in the installation—one for each member of the staff in a small or medium installation, and one or two for each group in a larger installation. The copies of the manual should be serially numbered, *1* of *75*, *2* of *75*, etc. so that the number of manuals and each person to whom one is assigned will be known for maintenance purposes.

The format of the manual has been previously discussed. It should be typed and reproduced if more than ten copies are required. Illustrations can be drawn and photographed, or drawn directly on multilith mats. In very large installations where more than 100 copies may be required, an inexpensive offset process may be used for printing and copy prepared by varityping or typesetting.

Step 7—Staff Training

The most important step in the installation of methods standards is proper training of the staff members in new procedures. The following should be included in the educational program:

a. Distribute the copies of the Manual of Standards along with a memorandum from the Department Manager requesting that each staff member read it and submit comments and suggestions.

b. Hold a general meeting with all of the interested staff members and discuss the manual, its concepts, and the suggestions that have been received.

c. Modify the manual, if necessary, to incorporate the suggestions made by the staff.

d. Hold a series of classes for each section of the department to acquaint all staff members with the new procedures. These can be held in a short period of time, since much of the information will have been read in advance. At this point all suggestions should have been incorporated or discarded, and no further change in the manual should be allowed, except through a normal review procedure.

Step 8—Prepare for Installation

A firm date of installation should have been established. Certain preparatory actions must be taken. These include design and ordering of necessary forms and the development of a formal procedure for review, maintenance, and up-dating of the manual.

Review.—Before the effective installation date the installation committee acts as reviewer of all suggested changes or additions. Since this committee is composed of all line supervisors, it is impractical to continue its use after installation. A formal review committee should be established, with representation from each affected area and the original members of the standards team. Staff members should be encouraged to make suggestions for improvement of the standard practices at any time and such suggestions will be reviewed by the Review Committee at least once a month. All suggestions will be considered on merit alone, and the suggestor will receive a written notice of the disposition of each suggestion, with a copy sent to the data processing manager. This should prevent morale problems which could be caused by management's failure to listen to employee suggestions for improvements in standard practices.

A number of suggestions will be received immediately after the initial installation has been made. Many of these will not be con-

structive, since they will represent the normal resentment to change. Some will be good suggestions and others honest representations of conditions where the designed standards do not work effectively or cause undue hardship.

Step 9—Installation

The actual installation of the new standard practices and procedures is not difficult. More difficult will be to insure that the standards are observed if they are of benefit, and changed when they are not effective. The installation date should be set taking into account the current workload conditions, since departmental efficiency will be somewhat reduced while the new procedures are being learned. The following is a typical installation schedule:

August	The data processing budget for next year is approved with an item reading: Standards development $20,000.
January 2	The data processing manager appoints two men to the standards team—a full time assignment starting on January 10. The team is advised that an outside consultant will spend two days per month reviewing their progress.
January 9	A memorandum is sent to all staff members inviting them to a meeting on standards on January 29 and explaining the purpose of the meeting.
January 29	The anouncement meeting is held; the executive vice-president opened the meeting, and expressed his personal interest in the topic—indicating that the company felt computers would ultimately handle most of the important functions and that Data Processing must be equipped to handle this responsibility. A questionnaire prepared by the standards team is handed out, with a returning deadline of February 9.
February 9	The questionnaires are mostly returned, with some meaningful suggestions, and some less constructive suggestions.
February 15	All of the questionnaires are in.
February 25	The standards team has prepared a complete Table of Contents.
February 28	The supervisors meet to approve the Table of Contents.
March 4	The Table of Contents has been approved; multiple copies have been made, and distributed to the staff.

May 7	The first draft of the contents is sent to be typed.
May 15	The supervisors meet to review the contents.
May 17, 19, 22	The supervisors and the team continue to meet.
May 27	The first draft, with changes, is approved.
June 24	The first edition has been completed.
June 26	Copies of the first draft are distributed with an invitation to all members for a staff meeting on July 8.
July 8	The staff meeting is held, the standards are explained; numerous suggestions flow in—some are again more constructive than others.
July 15, 17, 19	Classes are held by the standards team for the benefit of anyone with questions about the new procedures.
July 26	An installation date of September 30 is established, based on the projected work schedules for October.
August 1 to August 15	The installation committee meets to consider changes.
August 30	The final edition is made ready for typing.
September 5	Final forms designs are ordered.
September 10	Multiple copies of the 2nd revision are distributed.
September 16	A review committee is established.
September 18	Minimum standards for existing systems are published.
September 30	Installation is made.
October 4	The review committee meets to consider problems.
October 10	The third revision is sent to typing; the standards team is reassigned.

CHANGING EXISTING PROGRAMS OR SYSTEMS

A frequent question is the necessity of upgrading materials prepared before the installation of the new procedures. Is it necessary, for example, to revise or upgrade the documentation for currently operational programs that were completed long before the new standards go into effect? This question should be given very careful consideration.

Making the necessary changes to existing systems documentation and program manuals is extremely expensive, often prohibitive. If changes are not made, however, the total operating chain will be weak and existing programs will still be difficult to change and inefficient to operate and will in general encourage the continuation of those practices the new standards are supposed to eliminate.

The best answer to this question lies in the middle ground between changing all programs and not making any changes. A "minimum" standard for existing systems can be established which lies somewhere

between no standard and the ultimate new standard. All programs or systems whose documentation or methodology does not meet the "minimum" standard must be changed; those programs with the "minimum" necessary materials need not be changed. The depth of the "minimum" standard governs the cost of making the changes and the number of programs which must be changed. To insure a maximum of uniformity at the lowest possible cost the minimum standards must include the following:

Systems Documentation.—All existing systems should have the following minimum documentation:

- Title page
- Revision page
- Table of contents

(The above are easy to construct and lend an air of uniformity to the entire manual.)

- General description
- Card layouts
- Printer or form layouts or sample reports
- Flowchart of the new system

(These should already be available in some form. It is necessary only to collate them into a manual, and perhaps add a few paragraphs of general description.)

Program Documentation.—All existing programs should have an operating manual which contains minimally the following material:

- Title page
- Revision page
- Table of Contents
- General description
- Macro-block diagram
- Basic operator instructions or a set-up sheet
- List of halts and actions to be taken

All programs should also have an up-to-date symbolic entry deck, and a current listing with no more than five machine-language corrections.

Although the coding standards presented in Chapters IV and V are an extremely effective method for reducing the problems of program take-over, it rarely pays to go back and re-code programs written before the standards went into effect. It does pay to update the documentation to a minimum standard, as indicated above, but the complete changing of a program is warranted *only* if major changes are due to be incorpo-

rated into the program. It then becomes extremely economical to revise the remainder of the program to the new standards, because reassembly and testing must be done anyway, and because the person making the major change must become completely familiar with all aspects of the program. In these cases the following rules should be observed:

1. All program documentation should be revised to the minimum outlined.
2. Programs should not be recoded to the standard format unless

- It is economical to rewrite the program to optimize efficiency
- A major change must be made to the program in any case
- The failure to meet existing standards interferes with operation.

3. If a program is recoded for any of the three reasons above, the documentation should be revised from the minimum standard to the normal standard.

As a result, all of the systems and programs will have the basic minimum documentation within a very short period.

STANDARDS ENFORCEMENT

Standard methods and procedures lose their value if not practiced consistently in all areas of application. Once standards have been installed, policing is required to insure that they are adhered to. In the strictest sense this is disciplinary enforcement, which must be done using all of the available techniques of punishment and reward.

General Methods of Enforcement

Good standards enforce themselves up to a point. Their benefits are readily recognizable to the user and he will continue their use.

Enforcement Requires Strong Management.—The attitude of every level of management towards the establishment and enforcement of standards (as towards the ultimate role of data processing) must be positive and strong. The line supervisors constitute the first line of defense against the "encroachment" of poor working habits. They must continuously emphasize the importance of standardization, and rigidly enforce the standards. They must anticipate and resist the impulse to evade or avoid standards and almost unquestioningly adhere to the rules. The second line of defense is the general supervisor or data processing manager who must prevent the formation of conflicting interest groups. Many of the programming standards are for the benefit of

the operations group at a cost to programming. The manager must prevent the programming group from downgrading the value of these standards and, in general, each group is responsible for some rules which benefit other groups. Each group *must be forced* to recognize the *overall* benefit, by the data processing manager. Corporate management is the third line of defense. Corporate management must itself continually emphasize and demonstrate the value of standards as it does in other areas of the company such as engineering, manufacturing, and statistical analysis. The success of the standards program is often directly proportional to the strength of management and its ability to enforce its own demands.

Enforcement Requires Incentive.—The staff must recognize not only the value of the use of standards in making their work more meaningful, but also its value to their career advancement. This can be done in two ways: the first is to demonstrate to the staff that the quality of their work is being improved by the use of standards—that they are producing more and better work and will therefore advance more rapidly. The second way is to indicate satisfaction by praise and through merit increases, and to note that the adherence to standard practices contributes to advancement. Conversely, members of the staff who resist the use of the new procedures should be censured, and their failure to earn advancement should be blamed at least in part on their failure to follow the required rules.

Good Enforcement Requires Recognition of All Points of View.—One important aspect of the continuing standards program is the Review Committee and the review procedure. It is not possible to enforce standards if the staff does not have the opportunity for presenting its point of view on occasions when there are differences of opinion. All suggestions should be considered on merit alone; good ones should be adopted and poor ones should be explained to their author.

A Competitive Spirit Should Be Instilled.—One of the most powerful incentives to performance is the spirit of competition.

In one installation with which the author was associated it was a standard practice to levy a small fine (varying from 2¢ to 50¢) for infractions of the standard rules. The fines were contributed to a fund later donated to charity. This system included fines of 2¢ for a coding error, 10¢ for a console error, 5¢ for a nonstandard label and 50¢ for inadequate documentation. All staff members were extremely careful in adhering to the rules, because they did not wish to be embarrassed by being required to pay a fine. They were extremely eager to catch others in mistakes or infractions. This competitive spirit further improved the quality of the work. In another installation where programmers exchanged programs with each other for desk checking purposes they took a great deal of delight in catching each other's errors. Since they usually

exchanged with the same person, and the amount of time allowed was standard (and a function of program size) the number of errors that were caught kept increasing. Each programmer would count the number of errors found; his opposite number would then try to find at least as many or more. As an overall result, quality increased very rapidly in a short time.

Continued Education Is Necessary for Problem Recognition.—Analysts, programmers, and machine operators should recognize each other's problems in order to become fully effective. Just as programmers should be exposed to the problems of operation, so should analysts spend some time working directly with the programmers. More than that, all groups need external stimulation and continued education to advance themselves professionally and within the installation. In recognition of this and to promote the establishment of good standards, management should provide for continued advanced education for the members of the data processing department. Seminars or discussion groups on improved operating methods and techniques can promote standards enforcement. The fact that management recognizes the needs of the staff promotes goodwill; by allowing everyone an opportunity to discuss his own problems and to understand those of others, a good atmosphere is created for the installation and use of good practices.

Enforcement Checkpoints

The use of standard practices can be verified at specific points in each part of the department. These could be called *enforcement checkpoints,* i.e., specific points in the process where a sample is taken or a quality control test is made. These should be formally established and regular quality control procedures undertaken. Although the location of the exact point in the process may vary with each installation, the following are suggested:

Systems Analysis (In Process).—After the existing system has been completely analyzed, and before the new system has been designed, the work of the systems analyst is reviewed by his immediate supervisor or project leader. The main objectives of this first review are to provide

- An evaluation of the analyst's performance
- A review of the completeness of the analysis
- A guarantee that standards are being used

The main documents analyzed at this time will be the flowchart of the existing system, the general description of the system, and the document analysis. If errors are found or non-standard practices observed they can be discussed with the analyst and corrected before submission of the specification to the programming department.

Systems Analysis (Final Review).—When the analysis is completed, it should be reviewed again. Otherwise the analysis may be rejected for incompleteness by the programming group, in accordance with the standard procedure. This review must be fairly comprehensive, covering a rating of quality as well as an analysis of the standards used. The following areas should be rated:

- Adherence to standard practice
- Completeness
- Accuracy
- Clarity

Each of these categories is given a weight, and each is scored independently. The scoring may be mechanical, by counting the number of errors, or it may be evaluative. It may not be necessary to establish a formal point scoring system, but just after installing a new system of methods standards it may be quite beneficial.

The above four categories could be weighted as follows:

- Adherence to standard practice 40%
- Completeness 30%
- Accuracy 20%
- Clarity 10%

Each category is then scored independently, and the weights are applied to obtain the total score, which should be retained only by the systems manager. Dependent on the size of the installation, and the objectives of personnel quality evaluation, the rating system might be extended to apply to each element of the task, such as the flowchart or the input layouts. It should be remembered that this is an evaluation of quality and adherence to standard, not a performance evaluation. This topic is discussed in greater detail in Chapter IX.

Programming (In Process).—The first step in the programming process is a detailed evaluation of the job specification. Assuming that the specification has been accepted, the programmer next prepares the macro- and micro-block diagram. The first point of review is after completion of the micro-block diagram. The review could evaluate

- Adherence to standard 35%
- Neatness 10%
- Understanding of the problem 15%
- Logical completeness 25%
- Clarity 15%

The entire logic should be reviewed to insure that there are no inconsistencies or missing routines. Clarity, and the proper use of standard abbreviations should be checked by having the reviewer analyze the charts without the programmer present.

The second review checkpoint comes immediately before testing or program checkout and after the programmer or his immediate supervisor has completed the final deskchecking operation. Its major objectives are to:

- Guarantee adherence to standard labels and organization
- Insure that the coding corresponds to the block diagramming
- Insure that there are no obvious logical or clerical errors

Each of the three should be given the same weight; any violation or error which is found at this review should count the same, therefore. In order to properly enforce the established standards, any case where the programmer or analyst has used a non-standard practice must be redone.

The third evaluation checkpoint in programming takes place after testing and before final documentation. It may be skipped, in which case its functions are included in the final checkpoint described below. The objectives of the third checkpoint are to:

- Guarantee that the program is completely tested 60%
- Insure that standard testing procedures have been used 40%

The best approach is for the reviewer to take the micro-block diagram and the job specification, and create completely independent test data. The program is then run against the new test data. If the output is correct the program has been completely tested.

The second part of this review determines that the proper standard practices have been used in testing and test data preparation. This can be done by scanning and evaluating the Test Plan and Test Results forms and by constructing the desired statistics, as described in Chapter IX. One measure of testing procedure quality could be the number of "patch" errors in relation to the total number of in-memory corrections. Another could be the number of in-memory corrections to the total size of the program.

Programming (Final).—The last checkpoint in programming is the evaluation of the documentation transmitted to the operating department after completion of the program. This review should be primarily concerned with

- Adherence to standards 50%
- Clarity 30%
- Accuracy 20%

Several techniques can be used to evaluate documentation. The acid test that some installations have adopted is to give the documentation and input data to the most inexperienced operator in the installation. The operator is then required to set up, run, and monitor the job under all conditions. If the documentation for operating is accurate and complete the operator should have no difficulty, no matter what conditions arise. The remainder of the programming documentation is reviewed by the supervisor for completeness and adherence to standards. One method of insuring that no one forgets any part of the required documentation is to provide a transmittal check list, such as is illustrated as Figure 7-2.

Operations.—Measuring the observance of standard practices in the operating department is largely a matter of direct review. This is generally done in two ways; the first is to evaluate the records in order to determine the presence of any trend in percentage of set-up, percentage of rerun time and other factors that may point to a loss of control. The second method is to observe the operation directly, at unannounced intervals, to determine the methods actually used, and to see whether the operator's instructions are being followed. The former method is described in more detail in Chapter VIII; the latter is a matter of good supervision. More often than not failures at the operations level are rapidly brought to the attention of supervision by the fact that reports are not produced on time, that productivity of the department is reduced, and that overtime is increased.

MAINTENANCE OF THE STANDARDS MANUAL

After the installation has been completed and the standards are being enforced successfully, there is often a tendency to relax. This may have the effect of gradually reducing adherence to the standards. In order to avoid this, at least one person must be assigned permanent responsibility for maintenance of the standards manual. If a change is approved by the Review Committee and becomes effective, the person responsible should issue copies of the change to all members of the staff who have been assigned a copy of the standards manual. The revision page of each copy also must be noted and the master copy of the manual kept in the program library must be updated with a complete record of the change, its reason, and its effective date.

All changes made to the standards manual must be evaluated by the Review Committee. One reason for doing so is to determine whether

TRANSMITTAL CHECK LIST

TO: PROGRAM LIBRARIAN

FROM: PROGRAMMER

I AM ATTACHING COMPLETE AND STANDARD DOCUMENTATION FOR PROGRAM NO.____, NAME

_____, SYSTEM _____ FOR RELEASE TO OPERATIONS ON _____ 19___.

INCLUDED IN THE DOCUMENTATION IS THE FOLLOWING:

- ☐ PROGRAM DECK- CONDENSED
- ☐ SYMBOLIC ENTRY DECK-CURRENT (ASSEMBLY OF _____19___.)
- ☐ LISTING-SAME DATE
- ☐ MEMORY DUMP
- ☐ FORMULAS OR PROGRAM ABSTRACT
- ☐ GENERAL DESCRIPTION
- ☐ DETAILED DESCRIPTION
- ☐ BLOCK DIAGRAMS
- ☐ OPERATOR'S INSTRUCTION SET-UP / TAKE DOWN
- ☐ HALTS
- ☐ MESSAGES
- ☐ LAYOUTS INPUT / MEMORY / OUTPUT / TAPE / PRINTER
- ☐ CONFIGURATION REQUIRED
- ☐ STANDARD TIME FOR SCHEDULING
- ☐ SAMPLE REPORTS-CARDS
- ☐ FLOWCHART
- ☐ FEATURES, CAUTIONS, MODIFICATIONS
- ☐ TEST DATA SET AND ITS OUTPUT
- ☐ REVISION PAGE
- ☐ OTHER_____

ALL ITEMS NOT CHECKED ABOVE MUST BE EXPLAINED:

ITEM REASON FOR ITS ABSENCE

REVIEWED BY _____ DATE _____19___

APPROVED BY _____ DATE _____19___

RECEIVED BY DATE

FILED:

ACCEPTED BY OPERATIONS: _____ BY _____

FIRST RUN DATE _____ BY _____

EVALUTION OF DOCUMENTATION: EXCELLENT GOOD FAIR POOR UNACCEPTABLE

SIGNED BY _____ OPERATIONS MANAGER

DATE RETURNED TO PROGRAMMING _____

Fig. 7-2. Transmittal Check List.

or not the change will require change in existing systems or program documentation. If the change only affects future programs it should be so indicated so that all existing programs or systems may be exempted.

CHAPTER SUMMARY

The successful installation of standards depends largely on the management attitude and thought that has gone into preparation of the standard procedures. The installation process follows these stages:

1. A standards "team" is created and assigned to the task.
2. The staff of the department is made aware of the project and suggestions are elicited.
3. A Table of Contents for the proposed standards manual is developed and approved.
4. The contents of the manual are developed.
5. Management reviews and approves the final contents.
6. The final draft is developed and printed.
7. A staff education program is started.
8. An installation date is set up and the necessary supplies ordered.
9. The standards are installed.
10. Minimum standards for existing programs are developed.
11. Standards are reviewed and revised.

The continuing success of the standards program is largely dependent on enforcement methods. These usually include:

a. Strong management
b. Use of incentives
c. Recognition of the staff point of view
d. Development of a competitive spirit
e. Continued staff education

Specific points of review for enforcement of standards and maintenance of output quality include:

Systems Analysis: After completion of the analysis of the existing system and after completion of the job specification

Programming:	After the block diagram is completed
	After desk checking
	After testing
	After documentation is ready for transmittal to the library
Operations:	Through review of trends in operating statistics
	By direct observation of the procedures used

Questions for Review:

1. Why is strong management necessary to enforce discipline?
2. What qualifications would you establish for a standards "team"?
3. Construct a questionnaire for obtaining suggestions from the staff.
4. Develop checkpoint procedures for evaluation of performance and adherence to standards for systems analysis and programming.
5. Develop a measurement system for rating programmers in relation to their performance.
6. What other enforcement procedures would you use in your installation?
7. Develop an education program outline for standards installation.

Chapter VIII

PERFORMANCE STANDARDS: EQUIPMENT

INTRODUCTION

Performance standards are yardsticks with which to measure operating performance. They provide management with control, and allow variances to be investigated and rapid action to be taken whenever performance strays from the expected path.

A distinction must be drawn between an estimate and a standard. An estimate for example, attempts to predict *actual* running time. A standard, on the other hand, states what that time *should* be. An estimate may be adjusted for later use when the actual performance is known. A standard theoretically is not adjusted; a major variance results in management investigation and action.

In data processing, much as in any manufacturing process, standards may be established for both equipment and personnel. The methodology is somewhat different. The equipment is self-controlled, and a variance from standard therefore does not indicate lower "equipment efficiency"; it may indicate some weaknesses in the program or lowered operator effectiveness. Similarly, it is difficult to use time study techniques to establish the standards for programming; the speed of creativity is almost impossible to rate.

It has been found necessary to develop special quantitative measures that can be applied to the functions of data processing. These measures, and a general approach to establishment of performance standards are discussed in this and the next two chapters.

Classical cost accounting allows only three methods with which standards may be established. These, in order of preference, are:

- Time and motion study
- Study of past performance records
- Estimates based upon experience and judgment.

The normal concept of standard costs is applicable only to production

processes; never to "job shops" or other highly variable processes. The data processing operation, often considered related to a job shop, is thus left without an acceptable methodology.

Luckily, a fourth method of establishing standards exists, which uses estimates which vary based upon the parameters of the specific operation. Thus, machine processing time varies with the basic parameter of volume, program coding varies largely with the program size, and the block diagramming effort varies with program size *and* logical complexity. This method, along with evaluation of historical records, is the method of measurements outlined in this and the following chapters.

GROUND RULES

The major advantages of establishing accurate performance standards are that they

- Supply management with basic cost information
- Aid in controlling costs
- Facilitate budgeting
- Allow reasonable accuracy in equipment and resources scheduling
- Facilitate personnel performance evaluation

Basic ground rules are required to provide the correct environment for the establishment and use of performance standards, among which are the following:

1. Methods must be completely standardized if performance standards are to accurately reflect prevailing conditions.
2. The standards program must have the complete support of top management. Management support has already been emphasized in connection with the establishment of methods standards. The same arguments can be applied to performance standards.
3. The program must have the complete understanding and cooperation of the entire staff. This need has been demonstrated before; in the present case it will be less difficult to obtain if it is pointed out that a true measure of productivity used in evaluation will be directly reflected in compensation.
4. Rules to control quality must be established and enforced along with measures of quantity. Otherwise a tendency may develop for slower workers to increase output by reducing quality. It would be possible, for example, to turn a program over to production without thorough testing of all the possible conditions. This would reduce the total time necessary to develop the program at the expense of errors in production.
5. Accurate records of performance must be kept.

GENERAL APPROACH

The management control cycle discussed in Chapter II depends largely on the feedback of information. Similarly, the initial development of standards does depend on the accumulation of historical information. The general cycle appears as follows:

1. Development of the initial standard. On the basis of estimates, judgment, experience, or a quantitative measure based on evaluation of *operating parameters* (as discussed herein), initial standards are developed.
2. Schedule development. A schedule is established on the basis of the initial standards.
3. Gathering information. Careful records are maintained on actual performance. Analyses are made of performance against the schedule. Variances are determined and their possible causes established.
4. Taking action. Action is taken to account for each variance initially encountered. If a particular variance is consistently in one direction, without apparent explanation, the standard may be wrong and need adjustment. Otherwise action is taken to adjust performance such as the building of incentives, modification of methods or increase in supervision.

A standard should not be adjusted because of adverse experience based on one operating group or one sample. Such a standard should be adjusted only on the basis of consistent variance verified using a control group or other installation.

PROGRAM PARAMETERS

Practical industry experience indicates that the only meaningful parameter that can be applied almost universally in computing compiler time, for example, is program *size*. This assumes that the average number of macros, pseudo-operations, comments, or compiler-control entries will be reasonably constant for the installation. This is true in all installations using methods standards of the types outlined in Chapters IV and V: the number of comments will be dictated by the rules on program organization and the number of macro-instructions will be a direct function of the standard sub-routines and of the programming rules dictating the particular macros to be used.

The unit in which the parameter is expressed is of little significance: it matters but little if it is in number of cards or inches of symbolic deck. However, since the parameters of a program must be estimated before the program is actually written, it is important that the unit chosen permit accurate measurement. Consequently, the unit that lends

itself most easily to such estimation is the number of pages of coding anticipated, generally divided by 10 to facilitate handling. The following scale is suggested and is used throughout Chapters VIII and IX:

Number of Pages	*Scale Unit*
01 - 19	1
20 - 29	2
30 - 39	3
40 - 49	4
50 - 59	5
60 - 69	6
70 - 79	7
80 - 89	8
90 - 99	9, *etc.*

The first program parameter is therefore estimated size, determined before the program is actually written. (It has to be pre-determined, if block diagramming and coding performance standards are to be derived at the same time.)

The second program parameter is complexity—a subjective value which can be estimated in advance by the most experienced programmer or the program supervisor. The code used for this parameter in this book uses a scale of 6 possible complexities, ranging from simple to impossible:

A	Simple
B	Moderately Difficult
C	Difficult (Average)
D	Quite Complex
E	Extremely Difficult
F	"Impossible"

The last item on the scale is generally reserved for the one or two 'monster' programs that have been built up over the years; their size and complexity are such that it is easier to regard them as outside the range of estimate; each should be estimated by itself.

In establishing a complexity code for the programs to be written, two factors should be clearly kept in mind:

- There is no direct relation between complexity and size; size is separately estimated. The logical complexity is strictly a function of the type of program and the number of different conditions accounted for. Of course, a truly complex program would usually require a sizable number of instructions to handle all conditions. There are, however, a number of extremely complex programs,

such as tightly optimized subroutines, whose size is 1, yet whose complexity is D or E. Conversely, an extremely simple printer routine on a non-alphabetic machine may be quite lengthy because of editing requirements.

- The same person should establish program complexity in all cases. This will provide a truly comparable evaluation.

The third parameter affecting development and operating time is the number of input-output units used. An extremely large and complex program may use only one tape for input and one for output; the set-up time for this program will be considerably less than for a simple, small program which uses the printer, 6 or 7 tapes, and an on-line card reader. This parameter is called input-output complexity, and is a simple count of the number of input-output units used. It can be obtained by a rapid analysis of the flowchart.

Each program will therefore have three parameters, expressed as *X N/Y:*

X is the rating of complexity (A through F)

N is the number of pages of coding, divided by ten (01 through 30)

Y is the number of input-output units (01 through the maximum units)

These three parameters can be used to quantify almost every one of the values required for a program. The remainder of this chapter, and Chapter IX will deal with the establishment of standards using some or all of these and other parameters.

DEVELOPMENT OF EQUIPMENT STANDARDS

As seen in Chapter VI, some kinds of computer use are chargeable and others non-chargeable for rental purposes. The chargeable uses are generally:

- Productive time
- Assembly or compile time
- Testing time
- Rerun time: Operator error
- Rerun time: Program error
- Rerun time: Data error
- Demonstrations
- Training

Non-chargeable time falls into these categories

- Set-up
- Assembly set-up

- Testing set-up
- Scheduled maintenance
- Unscheduled maintenance
- Rerun time: Machine failure
- Rerun: Manufacturer's software error
- Utility failure
- Idle time
- Tape testing

Although most installations assume that there is no cost attached to the "non-chargeable time," this is a fallacy. One cost attached to non-chargeable time is the cost of labor for computer operations and peripheral functions such as tape librarian, air conditioning mechanic, etc. A second cost is the overhead of the extra operation, which may be considerable if the extra time forces overtime or the addition of another shift. And last, but not least, is the fact that ultimately the computer use will exceed the total available time. Whether or not the machine is purchased or the manufacturer charges for set-up time, when the total of chargeable and non-chargeable time exceeds 24 hours in a day, the added cost incurred will be that of a second complete computer.

It would therefore not be sound management practice to establish standards only for chargeable time. If this were done, the effects might be extremely efficient chargeable operation, at the cost of sloppy and inefficient set-up time, increases in overtime and extra shift operations, and the like. The standards suggested therefore apply to all categories of machine time; the assumption made is that the equipment is owned and not leased, so that all time is chargeable.

Standards for Productive Time in A Business Application

In a business application, the productive or machine operating time varies almost directly with known and measurable parameters. Thus, in a tape-limited system, the productive time is in direct relationship to the tape time, which in itself depends only on tape blocking and record length. For any given application, these factors are known in advance and may be calculated, so that the only day-to-day variable is volume, or number of records.

With manufacturer supplied programs, such as sorts, a general timing formula is usually made available and can be translated. The major variables which affect the calculation are again the file volumes, the record length, and blocking factors. Calculation of standard time is therefore fairly simple arithmetic once the parameters are known. Figure 8-1 shows the output of a computer program used to calculate the operating times for a series of programs. Programs numbered with an

S are manufacturer supplied sorts, and the sort formula factors are printed out, based on the input variables of V(Volume), B(Blocking), and CH(Characters). For direct productive programs, the tape times were calculated for each of the two channels of the input-output system. The total time in hours is based on the larger of the two channel times, shown in minutes.

Figure 8-1 shows calculations based on an assumption which may not always hold true: an "average" standard volume has been used, obtained by taking all historical volumes and computing the "average" volume. This may be useful in cases where the overall fluctuation in volume is not great. If this is not so, a different method must be used. One such method is to calculate the standard operating time for a series of widely fluctuating volumes, and then to draw a curve representing the entire universe of occurrences. A second method is to calculate the time, perhaps using a computer, based on a timing formula which holds the volume as a variable; times are then calculated for standard increments of volume which may be as small as 100 items or as large as 1000. As indicated in Figure 8-1, the timing of the run is proportionate to the total volume of information on one of the two *channels* available; any curve or constructed table of "time versus volume" would have to reference the total volume of the largest files mounted on *one* channel of a multi-channel system, or the total volume of all files on a single channel system.

Figure 8-2 is a curve which represents the relationship between time and the sum of the volumes of a master inventory file and a transaction file, both of which are mounted on the same channel of a two channel system. Figure 8-3 illustrates the development of a table, which has been designed for an entire serial application of a payroll system. The significant volume in this instance is the size of the master file, i.e., the total employment, which is readily known at the start of the payroll period.

Similar relationships can be developed for compute-limited applications; the unit record time will have to be established on the basis of an individual timing of the instructions used by an average active record, and an average inactive record.

Standards for Productive Time in A Scientific Application

Productive time analysis in scientific applications differs from business applications in the following respects:

- A majority of the runs are compute-limited
- The volume, or number of cases, which is run varies greatly

THE DIEBOLD GROUP, INC. -LOCKHEED GEORGIA STUDY -FINANCIAL PROGRAMS - DETAIL LIST 0 8/24/62 PAGE 09

PROG#	SYS	R A T E	S I Z E	T A P E	7080-CPU-HRS PER USAGE	ANNUAL	PER CHANNEL 1 TP 1	TP 2	TP 3	TP 4	TP 5	TP 6	TP 7	1107 TAPE UTILIZATION CHANNEL 2 TP 8	TP 9	TP 10	TP 11	TP 12	TP 13	TP 14	OTHER 1107-TPS PRT TP	CARD TP	1107 PERIPHERAL PRNT HRS	CARD HRS	ESTIMATED 1107-HOURS PER USAGE	ANNUAL	CORE-MEMORY UTILIZATION IN 1,000 S 7080 CHARS	1107 WORDS
-JOB-																													
RPT-F2																													
4707	047	F	31/11		.5	13.0	2.6	.3	9.4		38.6	.4		1.2	1.6	9.5	.5	3.7	.6		48.0		5.0	.3	1.6	41.6	85.9	15.9
4708	047	B	06/05					.2	.3					.5													24.9	4.5
4709S	047		/01					820F		286C1		47D1		96C2		128D2			ADD		85CH		18L	144V	.1	2.6	.2	
4710	047	C	22/07		.2	5.2	.6	.5		.3	.4	.2				.4					1.2		.8				47.5	8.9
			TOTALS			20.8																150.8		7.8		46.8			
7775	048	C	08/05		.2	6.0																			.1	3.0	34.3	6.2
7776S	048		/01																								9.0	1.6
7777	048	B	06/03																								21.3	3.8
7787S	048		/01		.1	3.0																					.2	
7788	048	C	08/06		.2	6.0																			.1	3.0	48.6	8.9
7792S	048		/01																								9.6	1.7
7795	048		09/03																								28.2	5.2
8796S	048		/01		.1	3.0																					7.5	1.3
8797	048	C	04/04		.1	3.0																					27.9	5.0
8798S	048		/01																								9.0	1.6
8799	048	C	08/04					.1								.3	.1				.3		.2				39.8	7.2
			TOTALS			21.0																6.0				6.0			
4712S	049		/01					815F		283C1		44D1		39C2		121D2			ADD		80CH		18L	19V	.1	2.6	.2	
4715	049	D	15/09					1.1				.1				1.1		.5									50.1	9.3
4734S	049		/01					184F		294C1		94D1		63C2		258D2			ADD		165CH		17BL	12V	.1	2.6	8.4	1.5
4735	049	B	11/02													.2					.2		.1				27.1	5.0
			TOTALS																			2.6				5.2			
3752S	050		/01					820F		286C1		47D1		40C2		128D2			ADD		80CH		18L	27V	.1	5.2	.2	
3753	050	D	12/08		.1	5.2		.3		.3						.2	1.4	3.1	.2		1.4				.1	5.2	53.1	9.7
3754S	050		/01					80F		278C1		37D1		73C2		118D2			ADD		65CH		30BL	118V			5.8	.9
3755	050	B	05/04		.1	5.2		.2								.6	1.6	1.6			.6		.8	1.3			15.4	2.8
			TOTALS			10.4																41.6		67.6		10.4			
1746S	051		/01		.1	2.6		86F		286C1		47D1		40C2		153D2			ADD		85CH		45BL	3V	.1	2.6	11.4	2.0
1747	051	C	08/05		.1	2.6										1.8											64.0	11.7
1748S	051		/01					81F		283C1		44D1		39C2		142D2			ADD		80CH		45BL	33V			10.8	1.9
1749	051	C	11/07													.1					.1			.7			32.7	6.0
			TOTALS			5.2																		18.2		2.6			
4711	052	B	04/02					1.1								.4					.4		.4				11.3	1.9

Fig. 8-1. Computer Timing Analysis—Productive Programs. (*Courtesy*, Lockheed-Georgia Company, A Division of Lockheed Aircraft Corporation)

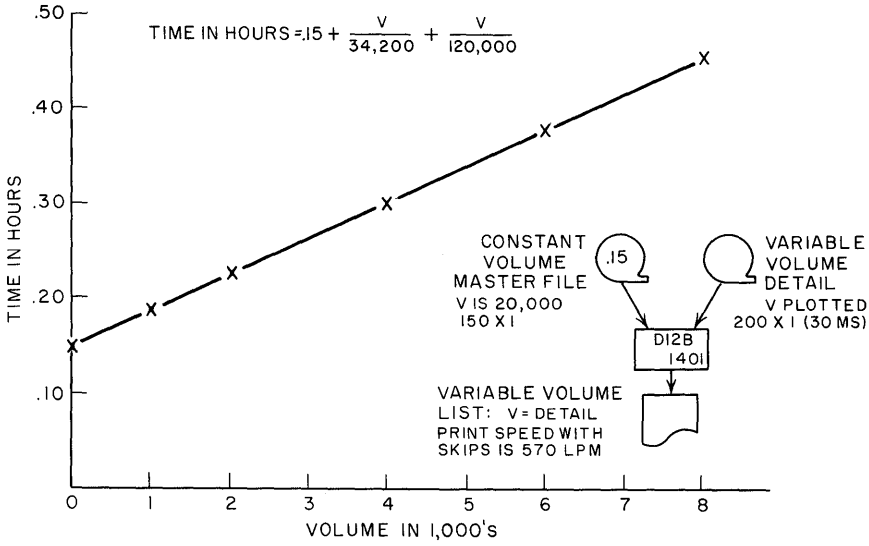


Fig. 8-2. Production Time—Standard Graph.

*CRITICAL VOLUME / PROGRAM	18,500	19,000	19,500	20,000	21,000	22,000	23,000	24,000
P01W	.16	.17	.17	.18	.19	.20	.21	.22
P02W	.10	.10	.10	.11	.11	.11	.12	.12
P03W	.62	.68	.73	.79	.88	.97	1.03	1.12
P04W	.20	.21	.22	.22	.23	.24	.25	.25
P05W	.10	.10	.10	.10	.10	.10	.10	.10
P06W	.30	.31	.32	.33	.35	.36	.37	.38
P07W	1.05	1.10	1.15	1.20	1.25	1.30	1.35	1.40
TOTAL	2.53	2.67	2.79	2.93	3.11	3.28	3.43	3.59

* CRITICAL VOLUME = 2 TOTAL EMPLOYMENT + 4 TOTAL NUMBER OF PAYROLL CHANGES (IGNORE IF LESS THAN 100)

Fig. 8-3. Production Time—Standard Table.

- There is no set "frequency" for most applications
- Some runs, such as Monte Carlo solutions, cannot be estimated in advance; their time depends on the nature of the problem and may vary from 10 minutes to 3 hours without advance indication.

As a result, it is more difficult to develop curves or tables of expected standard times. If volumes, or the number of cases, are known it is possible to actually use predicted times from an analysis of the program, by timing the "inner loop," i.e., that part of the program most frequently used. If the number of cases is not predictable, or the timing is not a direct function of tapes or an easily calculable inner loop, the best approach is to use empiric data. Based upon actual experience, it is possible to construct a scatter diagram which plots the total time per run against a time scale. This enables computation of "average" times, over certain time periods. It will enable the detection of any trend towards increase or decrease in total running time or the "frequency" of running. From this it is usually possible to construct estimated times which may be used for scheduling purposes.

STANDARDS FOR COMPILING OR ASSEMBLY TIME

There are two approaches to constructing a standard for performance evaluation of a compiling operation:

- A general approach that uses normal percentage of assembly to production
- A specific analysis to determine compile time per program.

In the general approach, the objective is to establish a standard percentage of compiling time in such a way that analysis of total machine usage indicates where total assembly or compile time has exceeded the norm. Thus, if the monthly machine utilization analysis showed the following:

Productive Time	154.3 hrs.	40.2%
Compiling Time	15.4 hrs.	4.0%

the analyst, or the manager would be able to recognize this as "acceptable" or "out of control," depending on the standard. The standard, therefore, can be a percentage of total time, or percentage of productive time, or total number of compiling hours that can be considered acceptable under normal operating conditions.

To establish this standard, it is necessary to evaluate the parameters upon which compile time depends. These are:

- Number of instructions per minute that the compiler or assembler can translate
- Number of programmers
- Number of programs produced per programmer

- Average number of compiles per program
- Average size of each program

To give an example: There are 10 programmers in an installation, 2 of whom are maintaining programs and 8 of whom are creating new programs. The maintenance programmers are responsible for the updating of 250 programs, which generate an average of 8 changes per week. Since a changed program is patched and recompiled alternately, the 8 changes require only 4 compiles per week.

New programs are created at a rate of 2 per week. As an average, each new program requires an initial compile, a final compile, and one during testing. The total compiles required by the productive programmers is therefore 6 per week. The average size is 6,000 instructions, and the manufacturer supplied compiler operates at an average speed of 400 instructions per minute.

The standard monthly compile time in hours would be calculated as follows:

$$\frac{\text{Number of compiles} \times \text{Average program size}}{\text{Compiler speed in minutes} \times 60}$$

Since 44 compiles per month are required, the result is:

$$\frac{44 \times 6{,}000}{400 \times 60} = 11 \text{ hours total compile time}$$

In the above example a variance in compiling time from the standard would be difficult to trace since:

- An average program size is assumed
- An average compiler speed is assumed
- An average number of changes is assumed
- An average of 3 compiles for each new program is assumed
- An average rate of new program production is assumed

Nonetheless the approach is usable, especially in large installations with many programs, where any "average" tends to become more realistic.

An approach which makes possible accurate analysis of variance and pinpoints individual responsibility is to develop a specific standard for compiler time required for each program. To do this, it is necessary to determine those program parameters which contribute to total compiling time. This may be done as outlined below.

Standards for Compiling Time—By Program

After a program has been rated, it is possible to establish standard compiler time. The following example shows the method used in developing a standard for the number of compiles and the time required for each compile. Both are significant: the former to determine the accuracy of coding and the number of clerical errors made in each compile, the latter to insure that machine operation is efficient.

Number of Compiles.—The number of compiles is a direct function of program complexity and program size. The following formulation expresses this number as a direct function of complexity and size for a typical medium machine system with a symbolic compiler:

Size and Complexity Rating	*Number of Compiles*
A 01 to A 09 B 01 to B 04	3
All other A B 05 to B 10 C 01 to C 04	4
All other B C 05 to C 08 D 01 to D 04 E 01 to E 02	5
All other C D 05 to D 09 E 03 to E 06	6
All other	7

On the same system, the standard program compile time is shown below:

Program Size	*Compiling Time*
01 - 02	6 minutes
03 - 04	9 minutes
05	11 minutes
06	13 minutes
07 - 08	15 minutes
09	18 minutes

plus two minutes for each unit of size in excess of 9.

Thus, a program rated as

C 07 would require 5 compiles of 15 minutes each
B 02 would require 3 compiles of 6 minutes each
E 12 would require 7 compiles of 24 minutes each.

To calculate total monthly scheduled compile time it is merely necessary to add the compiling time allowance for all of the programs scheduled for compilation. If each programmer records each compile and its time, the compiling allowance may be easily checked against actual. It is then possible to determine which programmers have a greater number of clerical errors or use more machine time than the standard allows.

A further refinement is to calculate the number of compiles required as a direct function of the *expected* errors. Chapter V outlined a standard for recompilation which stated that a recompile was required for each ten test shots. An initial compilation and a final compilation are also required so that the number of compilations is:

$$C = 2 + \frac{E}{10}, \text{ where } E = \text{Number of Expected Errors}$$

The number of errors, or test shots, must be estimated using a separate formula. The following formula has been used for the IBM 1401. The number of test shots required for any program is:

	Test shots added for level of complexity	
	3	A
	5	B
2 × Size Code *plus*	8	C
	11	D
	15	E

The total number of expected test shots or errors is the above value, less 2 for volume and systems testing.

This is explained in greater detail later.

Standard for Set-Up Time— No Monitor Control (Business Application)

The standard time required to set up a machine system that does not operate under monitor control is a direct function of the number of equipment units to be set up. There are two approaches to development of this standard:

- The general approach
- The specific approach, by program

The general approach standardizes set-up time as a percentage of total productive time. Average set-up time must be determined. This can be done by taking the total of all the input-output ratings and dividing by the total number of programs. Thus, if there are 250 operating programs and the total value of all the input-output complexities is 1475, the "average" or mean of the input-output units is 5.9. If the formula for set-up time in minutes for this system is one minute plus .8 times input/-output, then the average set-up time per run will be 5.72 minutes, or .095 hours. In an average 8-hour day, the computer processes 21 runs, accounting for 5.32 hours of productive time. Set-up time can then be calculated by multiplying 21 by .095, or a value of 1.995 hours. The average daily set-up percentage is the average daily set-up divided by the average daily productive time:

$$\frac{1.995}{5.32} \text{ or } 37.5\%$$

The better method, again, is to establish a standard set-up time for each program, retained with the standard for productive time. Using this, the total daily productive schedule can be established, and performance evaluated.

A specific standard for a given program is a direct function of the input-output units attached to the system which must be prepared for the program. For this method a standard set-up time is established for each unit and this applied to the number of units used. Standards for set-up which could easily be established using the classical techniques of time and motion study, can be derived as follows:

Tape Set-Up

	Tape mounting	*Tape removal*
IBM 7330	.8 minutes	.4 minutes
IBM 729	.6 minutes	.5 minutes
IBM 727	.6 minutes	.5 minutes
Univac III	.6 minutes	.4 minutes
Univac IIa	.8 minutes	.5 minutes

Card Reader Set-Up

Without file feed	.3 minutes for each 750 cards
With a file feed	.5 minutes total

Printer Set-Up

1.0 minutes without exact alignment
1.5 minutes with exact alignment using programmed line-up

General Console Set-Up

Dependent on console complexity .3 to 1.9 minutes

Rewind and Reel Change, Without Tape Swap

1.8 minutes average

Each installation can develop its own formula for determining a program set-up standard. For example, a set-up formula for an all-tape 7090 might be 1 minutes plus (.6 times number of tapes) or $1.0 + (Y \times 0.6)$. If take-down is to be included the same formula would be $1.0 (Y \times 1.1)$. Whether or not take-down is included can depend on the number of tapes. In a 12-tape installation a program with Y equal to 6 would leave six tapes available for overlapped set-up; take-down need not be included in this case and, if the next program required 6 or less units it would require no tape set-up. It is therefore a function of the scheduling program or the scheduler to determine the allowable standard, based on the maximum standard obtained from the formula modified by the reduction possible through overlap. The beauty of using a computer program to establish the schedule *and* to set-up the daily or weekly standards, is that overlapped set-up and all other pertinent variables can be directly programmed to achieve the best possible machine utilization. This is obviously not possible if the program parameters and the necessary formulas have not been developed.

Standard for Set-Up Time under Monitor Control

Many installations, especially those engaged in scientific data processing, use a monitor program to assist in program loading and scheduling. The main reasons why this is more feasible in a scientific installation are:

- There is little need for file retention, except for the output
- Many of the problems are extremely short; the input and the output can both be on common tapes

When a monitor is used in this environment, a program tape is prepared, followed by a tape containing all of the input for the series of programs to be run. The only set-up necessary is the mounting of the two input tapes and the console set-up required for each run. In addition, "free" or "scratch" tapes must be mounted on all units, because many runs will use work tapes in their calculation.

In this case the set-up time is largely proportional to the number of monitor "runs" made. Since the scientific installation is often a "job" shop, this factor frequently depends on the delay between getting a job and running it. By evaluating historical records, it is possible to standardize both the number of monitor runs and the number of programs each monitor run will use. A standard set-up can then be derived by adding the total console time for each program to the tape set-up and take-down time required by each monitor run.

Standard for Assembly or Compiler Set-Up

The set-up time for a compiler run has very little in common with the standard for the program to be assembled. It is a direct relationship to the number of input and output units used by the *compiler,* and it will be the same for each and every assembly. As a result, it is quite possible to establish an exact set-up time for each compiler operation; however, many compilers have the capability of processing more than one program in a pass. The standard compiler set-up therefore should be modified by dividing it by the average number of programs compiled in one pass.

A simpler approach when using a multi-program compiler or assembler is to set aside a given time each day when all programs to be compiled will be run. This means, in effect, that there will only be *one* compiler set-up per day, which can then be used as the standard. If the compiler uses 6 tapes only, and its set-up time is $1.0 + 6.0 \times .6$, or 4.6 minutes, the monthly standard for compiler set-up should be 1.69 hours.

Standard for Testing Set-Up without a Testing Monitor

The standard used for the set-up time required for testing, if no test monitor is used, will approximate the set-up time for the same program after it becomes operational. The same units must be set-up except that in testing the program is usually in card form rather than on tape. Additional time must be considered set-up for the loading of the program cards, normally a much larger deck than the final deck, since it may be composed partially for machine language (one per card) corrections.

If a *general* approach is used in computing the set-up to productive time ratio, the same method may be used to compute testing set-up. It may be possible to use a ratio of testing set-up to total testing time or, if it is possible to find a direct relationship between total testing and total production (this seems somewhat unlikely), a ratio may be used between the testing set-up and the total production. The general approach requires assumptions to be made for the following:

- Number of programs tested in the period
- Average number of test shots per program
- Average set-up time per program test (calculated by summing the total input-output complexity factors, and dividing by the number of programs).

The set-up time standard can then be calculated as:
No. of programs $\times$ No. of test shots $\times$ Average set-up time. The ratio of test set-up time to total testing can be calculated as

$$\frac{\text{No. of programs} \times \text{No. of test shots} \times \text{Test set-up time}}{\text{Total test time}}$$

which should equal the ratio

$$\frac{\text{Average test set-up time}}{\text{Average test time}}$$

If a specific approach is used, the testing set-up time for each program is estimated in the same way as when the program becomes operational. To compare total set-up with the standard in any period it is necessary to calculate the set-up for each program, and multiply it by the number of tests performed. If the number of tests performed is in excess of the "standard" number of tests allowed, or if the standard allowable tests have been performed in preceding periods, a second variance is developed:

$$\frac{\text{Standard test set-up time for the program} \times \text{Number of tests performed}}{\text{Actual test set-up time for the program}}$$

which equals "set-up efficiency," and

$$\text{Number of tests performed in excess of standard} \times \text{Standard set-up time}$$

is equal to the set-up variance caused by excessive (ineffective) testing.

In any case, some caution must be used in the evaluation of test set-up. If the manufacturer does not distinguish in rental charges between test set-up and test time they can be combined much more effectively in one standard, i.e., X minutes per test for a program of a given size or complexity. Since a good part of the test session consists of manual intervention, set-up for personnel performance evaluation is not the same as "set-up" for chargeability—which includes all manual interventions.

Standard for Testing Set-Up with a Monitor

If a monitor is used to control the test, provide automatic program loading, generate required test data, and provide linkage to the memory and tape print out programs, the set-up time is considerably reduced. The standard set-up in this instance is that required to initiate the *monitor.* This is generally equal to the time required to

- Load the monitor tape
- Load a program and/or a data tape
- Load an output tape
- Set-up the console

Since a monitor usually provides the ability to run a number of tests in sequence, the total set-up calculation should include consideration of the manual time required to proceed from one test to the next as well as the total take-down time. As in multi-program compilers, the calculation can be made in two ways:

$$\text{Standard test set-up time per program} = \frac{\text{Monitor set-up time} + (\text{No. of programs-1}) \times \text{Manual time}}{\text{No. of programs}}$$

or, if the number of daily test sessions is kept constant:

$$\text{Standard monthly test set-up time} = \text{Working days per month} \times \text{Test sessions per day} \times \text{Set-up per session}$$

Standard for Test Time without a Monitor

Several alternatives are available for the development of a standard for program check-out time. These include considering test time as a ratio of productive time or total time, an equally general approach which uses average program test time, and specific calculation of the number of test shots required by each program and the length of each test shot.

Test Time as A Ratio.—If a great deal of program maintenance testing is done while new development testing is relatively constant, it may be possible to establish a direct relation between test time and productive time. This assumes that the productive time is somehow related to the number of programs under development and that the number of programs currently in use determines the amount of maintenance programming and maintenance testing required. Unfortunately, these assumptions are seldom valid so that a ratio between test and productive time is

usually somewhat spurious. If such a ratio must be used, the best is between test time and compiler time; in essence this reflects the number of new programs and the number of programs changed.

Test Time as An Average.—The test time standard established using this approach is a direct function of the

- Average time per test shot, and
- Number of test shots performed

The average time per test shot is determined empirically. The variance between installations is quite large: in a recent study of a 7090 installation with closed-shop testing, the author found the average duration of a check-out of all types to be 3.6 minutes. In a large 705 installation, with programmers at the console, the average total time per period was 13.5 minutes, broken down approximately as follows:

Set-up of test (no monitor)	3.0 minutes
Program Load	1.5 minutes (250 cpm, average program 350 cards)
Program Run	2.5 minutes
Intervention	3.0 minutes (including error look-up, transfers, type-outs, etc.)
Memory and Tape Print on Tape	2.0 minutes
Take-down	1.5 minutes

A large part of this time could have been eliminated through the use of a monitor and elimination of manual intervention at the console.

The number of test shots to be performed in a normal period is determined by analyzing:

- Number of programmers
- Type of changes or new programs produced
- Number of test shots required per change or new program

Assuming the following data, for example, for a one month period:

Number of maintenance programmers	5
Number of changes per programmer	7
Number of test shots per change	3
Number of programs finished by new development programmers	2
Number of test shots required for each program	16

Total number of test shots per month is $(5 \times 7 \times 3) + (2 \times 16) = 137$. If the average time per test shot is 8 minutes, the monthly testing "standard" is 18.27 hours.

Test Time Calculated per Program.—The number of test shots required for a specific program is related to program size, complexity, and number of input-output units controlled. The exact relationship is difficult to establish; the following typical values have been used by the author in practice and found fairly reliable. (All values assume closed-shop testing):

For the IBM 1401 Tape System:

	Test shots for level of complexity	
	3	A
	5	B
2 × Size Code *plus*	8	C
	11	D
	15	E

For the IBM 7080 (coded in COBOL):

$$\frac{1.5 \times \text{Number of generated instructions}}{280} \textit{ plus}$$

	Test shots for level of complexity	
	2	A
	4	B
1 for each tape unit used in excess of 4 *plus*	7	C
	10	D
	14	E

In this formula it is assumed that a page of coding generates approximately 20 machine instructions. The unit of size represented by 280 instructions is then equal to 14 pages of coding.

For the IBM 7090 (coded in FAP):

$$\frac{1.2 \times \text{Number of binary instructions}}{400} \textit{ plus}$$

	Test shots for level of complexity	
	4	A
	7	B
2 for each tape unit used in excess of 4 *plus*	10	C
	15	D
	19	E

For the UNIVAC 490 (coded in SPURT):

2 × Size Code (10 coding page units) plus

	Test shots for level of complexity	
	3	A
	5	B
2 for each tape unit used in excess of 4 *plus*	8	C
	11	D
	15	E .

Figure 8-4 displays in tabular form the IBM 1401 test shot standard. With this table it is quite simple to pick out the required number of test shots for each program, when scheduling total test time.

After calculating the number of shots required by the program, it becomes necessary to determine the length of the average test shot, as explained earlier, by empirical analysis.

TASK: TESTING

SHOWN: NUMBER OF TEST SHOTS PER PROGRAM
FORMULA: 2 X SIZE CODE
+
A = 3
B = 5
C = 8
D = 11
E = 15

SIZE CODE / COMPLEX.	01	02	03	04	05	06	07	08	09	10	11
A	5	7	9	11	13	15	17	19	21	23	25
B	7	9	11	13	15	17	19	21	23	25	27
C	10	12	14	16	18	20	22	24	26	28	30
D	13	15	17	19	21	23	25	27	29	31	33
E	17	19	21	23	25	27	29	31	33	35	37

Fig. 8-4. IBM 1401 Performance Standard—Test Shots.

The standard test time for a C-5 program in Figure 8-4, an average program, is calculated as follows. The program requires 18 test shots, distributed approximately as follows:

Program Testing	15
Production Testing	2
Systems Testing	1

The average systems test requires approximately 25 minutes of productive program time, the average production test requires approximately 35 minutes, and the average test time for a program test is 6 minutes. The total test time for the program rated C-5 is thus $(6 \times 15) + (2 \times 35) + (1 \times 25) = 185$ minutes. Since there are 18 shots performed, the average program requires 10.2 minutes per shot.

An alternate approach is to make test time per shot a direct function of either size or complexity. Program size is generally more appropriate than complexity, and the following typical relationship might be used:

Minutes per Test Shot	*Program Size*
6	1 - 3
8	4 - 6
10	7 - 9
12	10 and over

The prime purpose of showing the specific formulas which appear in the previous pages is to provide the reader with an *approach* that can be adapted to his requirements. The formulas and relationships shown have been used and validated by the author, but each installation must derive its own relationships, which take into account machine type, language, type of applications, and the general procedures used.

Standard for Test Time, Monitor Controlled

If a monitor is used to assist in the testing, the test time standard will be markedly reduced. Testing monitors are available for all commercial and scientific systems, and provide the following functions:

- Clearing of memory
- Rewinding of all tapes
- Generation of data, where desired
- Distribution of one set of data on all required input tapes
- Program loading
- Direct linkage to memory and tape print programs, or other diagnostic aids
- Multi-program testing without intermediate set-up

In calculating the amount of test time to be used for each program, the number of test shots will not be different from the standards used without the monitor. The only difference will be in the number of minutes required for each shot, which can again be derived empirically or based on program size or other criteria.

The actual test time used by a monitor-controlled system lies some-

where between 4 and 7 minutes per shot. The variation, if related to size, could be estimated as follows:

Minutes per Test Shot	*Program Size*
4	1 - 3 units
5	4 - 6
6	7 - 9
7	10 and over

If empirical data is used, it must include a large enough sample size so that the standard includes the proper number of production and systems tests.

Standard for Preventive Maintenance

Preventive maintenance is provided by the manufacturer under an agreement which should specify the exact period set aside for this function, its frequency, and the estimated length of time required. Such a schedule might read:

> The machine shall be turned over to Customer Engineering not less than two times per week, for a minimum period of two hours, to occur between the hours of 7 a.m. and 6 p.m. The time set aside normally will be from 7 a.m. until 9 a.m. on Tuesday and Thursday, unless a holiday intervenes, in which case Monday or Friday, respectively, will be substituted.

This, in effect, establishes a standard for the preventive maintenance function: 4 hours per week, or 17.3 hours per month. A variance on the low side would require management to contact the manufacturer to insure that sufficient maintenance is being provided. A variance on the high side, if warranted by the condition of the equipment, would reduce the total availability of the system for productive or other purposes. It is therefore quite important to set a basic standard in this area and insure that a variance is properly reported.

Standard for Unscheduled Maintenance

Unscheduled maintenance occurs when a machine failure is sufficiently persistent or catastrophic to warrant turning over the machine to the maintenance engineers for corrective action. Both scheduled maintenance and unscheduled maintenance occur on isolated units of the system and often do not require disabling of the entire system. If a tape drive is "down," or being cleaned as part of preventive maintenance, the system can operate on all runs that do not require the maximum number of tapes. Similarly, if the on-line printer fails, and an extra tape is available, all instructions to print can be changed to write the record on tape

instead. But, if the main frame (Central Processing Unit, Main Computer, or Console) fails, or a complete tape channel is disabled, the entire system must be turned over to the maintenance engineers.

To determine a realistic standard for "down" time or unscheduled maintenance, the frequency and average length of occurrence are used.

Frequency of occurrence can be obtained as Mean Time Between Failures (MTBF), which should be a part of the manufacturer's specifications for the system. The manufacturer has derived this value (which may vary from 10 to 1000 hours) on an empirical basis while the first few systems were being tested. The MTBF gives an effective frequency standard; if it is exceeded by the system, the system is not meeting its specifications.

The average length of down time is more difficult to obtain. When the system fails, the maintenance engineer administers diagnostic program tests, to determine the locality of the possible cause of the failure. In many instances, the failure will be found and corrected in a few minutes; in some cases it may take as long as a few hours, and in an isolated case, where a major part must be replaced and has to be shipped from the main plant, the down time can be as much as 24 to 48 hours. An average can be obtained from the manufacturer, however, based upon the aggregate experience of all the users of the same system, since the manufacturer's maintenance personnel is required to keep extremely careful records of down time. As a result, an average time can be used, and multiplied by the standard frequency.

Two variances can be obtained from this analysis. The first reflects a failure of the MTBF specification, and occurs if the number of failures exceeds the standard *consistently*. The second variance is based upon the actual time spent in unscheduled maintenance. This may be considered the effectiveness ratio of the maintenance engineering staff, and is very significant in proper management control. The major reason for this is the fact that the quality of the maintenance engineering staff is almost directly reflected in unscheduled maintenance. Several installations in the author's experience have had a completely unsatisfactory down-time record and replacement of the assigned engineer(s) was all that was necessary to rapidly improve the frequency and the length of unscheduled maintenance.

Standard for Utility Failure

Three basic utilities are required to operate most computer systems:

- Electric power
- Air conditioning
- Steam for humidity control

Failure of any one of these usually prevents extended operation, especially under adverse climatic conditions. Standards for utility failure may be difficult to obtain and, since a utility failure is fairly rare, should probably be expressed and evaluated on an annual basis, if at all.

Electric Power.—Many installations can make use of standby or home-generated electric power to maintain minimum operation in the event of externally supplied power failure. In this case, no standard need be applied for failure of electric power. If no standby power is supplied, some standard should be established for frequency and length of power failures caused by generating equipment failure, or in the case of outlying installations, by weather conditions or other problems. The factors that determine this standard should in general be obtained from the local utility company.

After obtaining the estimated standard for power failures, a record should be kept of actual occurrences. If the number or duration of power failures exceeds the standard to an extent that affects the efficiency or effectiveness of the operation, some consideration should be given to a stand-by generating system for the computer installation. Depending on the size of the installation, such a system can be obtained for between $10,000 and $40,000, with a small annual charge for maintenance and cleaning. They are generaly constructed with an automatic cut-in switch which acts in the event of a power failure or a severe drop in the line voltage. Immediate takeover is important, especially in the case of drum or disc systems requiring a long period to decelerate and accelerate.

Air Conditioning.—A failure of the air conditioning system does not have the immediate effect of turning the computer system off. Most computers are built with protective switching devices, geared to detect a rapid change in temperature. On some systems these devices turn off the machine if the ambient temperature reaches a certain point, on others the rate of change in temperature is the restrictive factor, e.g., no more than 4 degrees in either direction in an hour. A standard for air conditioning failure can only be obtained from the manufacturer, based upon the specified Mean Time Between Failure and the average down time (a partial function of where the maintenance engineer is stationed). A variance from standard, or excessive failure, may point to the need for a standby air conditioning plant. In this instance, it is not necessary to duplicate the entire system: if a 20-ton system is required to maintain operating temperature, a total of three 10-ton units will provide a standby of one 10-ton unit.

Steam.—Some systems use steam to maintain necessary humidity controls. The effect of a steam failure is less immediate than any other failure, but even this will be felt rather quickly. Alternates include portable humidifiers, which operate on a portable supply of water. Standards for steam failures are difficult to obtain; if in-plant steam is

used, the manufacturer of the generating system should be able to provide the necessary statistics. If not, the supplier should.

Standards for Rerun Time

Rerun time is usually that time lost because of an error. If an error occurs in the middle of a production program, the time lost is the time already expended. The program will be rerun, and the total time to run the job will be 1.5 $\times$ the normal time. In this case, the rerun or lost time will be ½ of the normal running time. There are four distinct causes of lost time:

- Data error
- Program error
- Machine error
- Operator error

It is important to retain the distinction between these, since at least three reflect upon the quality of personnel performance: the program error upon the programmer, the machine error upon the maintenance personnel, and the operator error upon the operator.

It is extremely difficult to set a specific standard for these four categories. As a result, a more general approach must be used. Since all reflect on operating quality, an effective technique is statistical quality control to review the percentage of failure as applied to total productivity, as follows:

$$\text{Data error, \%} = \frac{\text{Total time lost due to data error}}{\text{Total productive time}}$$

$$\text{Program error, \%} = \frac{\text{Total time lost due to program error}}{\text{Total productive time}}$$

Standards for the remaining causes of lost time are calculated in the same manner.

Each of these values should be plotted on a "control chart," designed in accordance with principles of quality control.* Figure 8-5 shows a typical quality control chart for the data error percentage. Since only one year's data was available, it has been used; normally a larger sample is required to use the technique accurately.

In the illustration a number of points are outside the normal quality control limits, based upon a normal, bell-shaped distribution curve. The

* A detailed description of the techniques and formulas used can be found in Grant, E. L. *Statistical Quality Control,* McGraw-Hill, New York, 1956.

lower points need no investigation in a process which attempts to minimize the values; the upper points that are out of control should be carefully investigated. From the notes on the chart it can be seen that the "out-of-control" points have reasonable causes; the absence of the supervisor and the replacement of operators on vacation.

Management control can be greatly enhanced by the careful control and record keeping procedures inherent in the retention of data for quality control analysis. If similar charts are kept on operator error, program error, and machine error, the basic causes for these factors will be rapidly uncovered and management will be able to apply corrective measures.

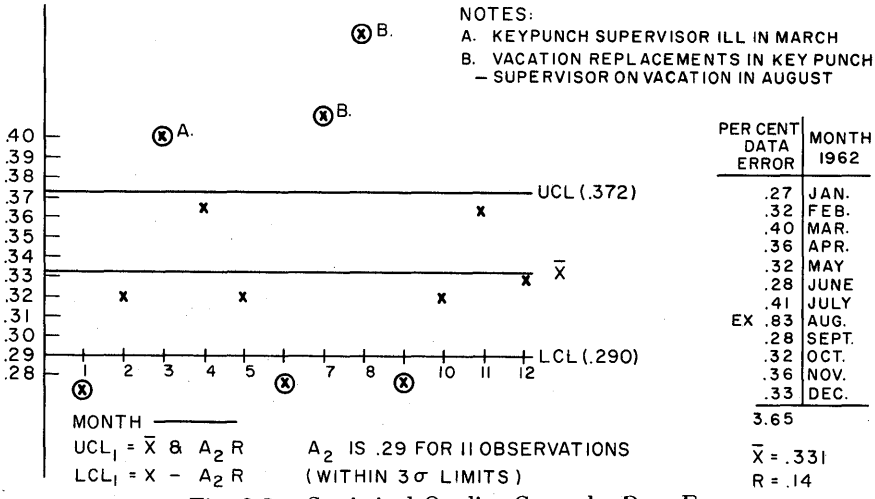


Fig. 8-5. Statistical Quality Control—Data Error.

Standards for Tape Testing

In some installations, especially those lacking a tape system with a "read-after-write" checking feature, it is common practice to test the available tapes for possible scratches and other error-causing conditions. A special tape testing machine may be used or the tape passed twice through the computer; large data records are written on the tape on the first pass and the records checked for accuracy on the second pass. If the computer is used, a standard should be established for this procedure as follows:

Total tape testing time in a period equals:

$$\frac{2 \times \text{Total no. of tapes in library} \times \text{Average tape passing time in minutes}}{\text{Interval between tests.}}$$

Thus, if there are 1,000 tapes in the library, each of which is tested every six months, with an average passing time (for 2,400 feet) of 4.5 minutes, the monthly standard for tape testing should be

$$\frac{2 \times 1000 \times 4.5}{6} = 1500 \text{ minutes, or } 25 \text{ hours}$$

This assumes that the tape test program overlaps tape rewinding, and that the system has only one channel. If more than one channel is available and used, the time is reduced by dividing by the number of channels.

Standard for Demonstrations

The novelty of electronic data processing makes it almost inevitable that corporate management will demonstrate the equipment to customers and other business associates. Many companies have specially designed demonstration programs, which type or print out, at high speed, the virtues of the company and its products. The time set aside for these demonstrations is normally charged to an overhead account. Establishing a standard for this is almost impossible, corrective action is equally impossible, and an arbitrary standard will suffice for record keeping purposes. If the standard is made low enough, a large variance will call excessive demonstrations to the attention of management, who will then perhaps reduce the number and length of the visits.

Standard for Training

The machine is used for training purposes whenever a new operator is given practice or a new programmer is allowed to operate the console. A training allowance should be established for each new person, based upon the machine's availability and the cost of machine time. If sufficient machine time is available as much as an hour may be used as the training standard for each new employee in operations and programming. To calculate the monthly standard, merely multiply the total number of new accessions in these departments by the time allowance.

Idle Time

All other machine time not charged to any other function is idle time. This time cannot be standardized, except as a negative factor; that is, by summing all the other standards, and subtracting this total

from the total available time in the period, the net idle standard can be obtained. In all accounting for machine time the idle time must be recorded, for control balance purposes. The idle time may also be plotted on a graph to determine machine use trends. On this basis projections of usage increases can be made and more or faster equipment ordered well in advance of the point when the total capacity is exhausted.

The methods for developing these standards, and the method in which the variance can be calculated is summarized in the Table on pages 234-235.

DEVELOPMENT OF A SCHEDULE

On the basis of the established standards, it is possible to develop a detailed daily schedule of work by either using a computer program to calculate each detail and determine all allowances, or by using a simple assignment sheet to apply the necessary standards and provide buffer times for occurrences of "non-scheduled" events such as re-run, utilities failure, and unscheduled maintenance.

Ease of schedule development is one of the basic advantages of good performance standards but a schedule is not essential to the accurate measurement of data. Nevertheless, the development of a schedule, or even of a sequence of tasks to be performed, provides extremely good discipline over the operating staff. Part of such a schedule is shown below:

			Standard Time in Minutes		Approximate		
Job Number	*Name*	*Shots*	*Set-Up*	*Run*	*Start*	*Stop*	*Check*
D17B	Category Sort		4.6	23	9:30	9:58	
D18B	Edit		2.2	11	9:59	10:12	
D19B	Update Master		3.5	17	10:13	10:34	
P12D	Labor Distribution		3.5	24	10:35	11:02	
B	Buffer		.....	20	11:02	11:22	
T	Test-L System	(6)	11.3	10	11:23	11:45	
A	Assembly-SLEUTH	(4)	3.3	16	11:46	12:05	
P01W	Gross to Net		4.1	42	12:06	12:52	
P02W	Sort-Employee Seq.		4.4	16	12:53	1:14	

It should be noted that the schedule form is not used to record actual time. This should be a separate function, with its own procedures and controls. The check mark merely serves to indicate operator compliance. Many installations require an operator initial in the same place.

MEASUREMENT OF EQUIPMENT UTILIZATION DATA

Chapter VI has outlined in detail the many techniques and types of equipment available for the accurate measurement of computer utilization and breaking it down into the many categories in which management is interested. The basic record obtained is a log record showing the time for each specific function, with category codes attached. This record may be obtained by the computer itself, by a recorder, by a clock, or by manually recording the elapsed time for the specific task.

ANALYSIS OF THE DATA

The computer may be used to prepare the necessary analytical reports, or the information can be summarized and tabulated by hand or by using punched cards. The major objectives of analysis are:

- To compare the actual performance in each category with the established standard
- To determine the effectiveness of personnel
- To account for and charge to the appropriate departments for services and to determine total rental due
- To determine trends, and recognize their impact on future data processing requirements
- To indicate management action where performance is not satisfactory, to optimize effectiveness of management policies, and to provide measures of the effect of such management action.

To Compare Actual with Standard

To compare the actual performance in each category with the established standard requires a summary of utilization by category, the calculation of the percentage distribution of categories, and the calculation of variance from the applicable standards. The calculation of variance may be in terms of frequency, and/or time, and may be expressed in percentage points, shown in the table on page 236.

SUMMARY OF EQUIPMENT STANDARDS

Category	*Approach*	*Parameters*	*Formulas and Relationships*	*Variance*
Productive - Business	Specific	Volume, Block, Length	$T = F$ (Volume, Blocking, Length)	Time
Productive - Scientific	General	Empiric	Historical - Average	Time
Compiling	General	Size, compiler speed	$T = \frac{\text{No. of compiles} \times \text{Average size}}{\text{Compiler speed} \times 60}$	Time
Compiling	Specific	Size, Complexity	$N = F$ (Size, Complexity) 2 Alternates	Number
Compiling	Specific	Size, Complexity	$T = F$ (Size), Σ by Program	Time
Set up - No Monitor	General	I/O Complexity	$T =$ Average S.U. as F (I/O) $\times$ No. of set ups	Time or %
Set up - No Monitor	Specific	I/O Complexity	$T = \Sigma$ S.U., by Program	Time
Set up - Monitor	General	Monitor I/O	$T =$ Monitor Set Up $\times$ No. of runs in period	Time
Compiler - Set up	General	Compiler I/O	$T =$ Compiler Set Up $\times$ No. of days in period	Time
Test Set Up - No Monitor	General	I/O Complexity	$T =$ No. of Progs $\times$ No. of test shots $\times$ Test set up	Time
Test Set Up - No Monitor	Specific	I/O Complexity	$T = \Sigma$ Set Up, by Program	Time
Test Set Up - No Monitor	Specific	I/O, Complexity, Size	$N =$ Number of test shots as F (Parameters)	Time - Excess SU
Test Set Up - Monitor	General	Monitor I/O	$T =$ Monitor Set Up & Linkage $\times$ No. of Days	Time
Test Time - No Monitor	General	Ratio of Total	$T = K\% \times$ Total Time	Time/%
Test Time - No Monitor	General	Number, Time	$T =$ Average time $\times$ Number of shots	Time
Test Time - No Monitor	Specific	Size, I/O, Complexity	$N = F$ (Size, I/O, Complexity)	Number
Test Time - No Monitor	Specific	Size, I/O, Complexity	$T = N \times T_i$ as F (Size)	Time
Test Time - Monitor	Specific	Size, I/O, Complexity	$N = F$ (Size, I/O, Complexity)	Number
Test Time - Monitor	Specific	Size, I/O, Complexity	$T = N \times T_j$ as F (Size; Monitor)	Time
Preventive Maintenance	General	Contract	$T =$ No. of hours, and frequency, specified	Time
Unscheduled Maintenance	General	MTBF	$N = F$ (M.T.B.F.)	Number
Unscheduled Maintenance	General	Empiric Data	$T = \Sigma$ Experience $\div N$	Time
Utility Failure	General	Empiric - Manufacturer	$N = F$ (Utility Experience)	Number/Year
Utility Failure	General	Empiric - Manufacturer	$T = F$ (Utility Experience)	Time/Year

Rerun - Data	General } Experience,		
- Machine	General } X and R		
- Program	General } within	$\%\ \text{Rerun} = \frac{\text{Time Lost}}{\text{Total Time}}$	%
- Operator	General } 3σ limits		
Tape Testing	Specific No. of Tapes, Time	$T = \frac{2 \times \text{No. of Tapes} \times \text{Tape Passing Time}}{\text{Frequency of Testing}}$	Time
Demonstrations	General Arbitrary	T = 1 hour/month Arbitrary Value	Time/Number
Training	General New Hires	T = No. of Training Hrs. × No. of New Hires	Time
Idle		T = 24.00 — All Other	0

After the calculations have been made, a one-page summary report is produced, showing the following:

	Actual			Standard				
Category	*Hours*	*%*	*Number*	*Hours*	*%*	*Number*	*Variance %*	*Type*
Production	143.7	38.4	312	129.5	37.1	312	+14.2	T *
Set-up	95.3	25.5	298	88.7	25.5	312	+ 6.6	T *
							−14	N *
Compiling	11.1	3.0	73	12.7	3.7	70	− 1.6	T
							+ 3	N
Compiler Set Up	1.5	.4	22	1.7	.5	22	− .2	T
Testing	24.7	6.6	134	21.1	6.0	120	+ 3.6	T
							+14	N *
Test Set Up	18.3	4.9	44	14.7	4.2	44	+ 3.6	T
•								
•								
•								
•								
Totals	374.2	100.0		348.8	100.0		+25.4	T

T = Time N = Number

This is the simplest form of report with which to compare actual performance against pre-established standards. Variances marked with an * are to be investigated and are the productive and set-up time, and the number of test shots. In this sample case, the number of set-ups showed a favorable variance (through the overlapping of several set-ups or the use of continuous serial runs, which had not been included in the standard calculation) even though the total set-up time was still in excess of the allowed variance.

To Determine the Effectiveness of Personnel

To determine the effectiveness of personnel as reflected in equipment performance, analyses can be made of the data presented in the above report. The following ratios can be derived for the purposes indicated:

Ratio	*Measures*	*Values from Report*
$\frac{\text{Standard Number of Set-Ups}}{\text{Actual Number}}$	Effectiveness of Operators	$\frac{312}{298}$ 104.2%
$\frac{\text{Standard Set-up Time}}{\text{Actual Set-Up Time}}$	Operator Efficiency	$\frac{88.7}{95.3}$ 93.5%

Standard No. of Compiles / Actual No. of Compiles	Programmer Care	70/73	96%
Standard No. of Tests / Actual No. of Tests	Testing Effectiveness of Programmers	120/134	89.5%
Standard Test Time / Actual Test Time	Testing Efficiency of Programmers/Operators	21.1/24.7	85.7%
Standard Number of Unscheduled Maintenance Events / Actual No. of Failures	Effectiveness of Maintenance Engineer	Not shown	
Standard Down Time / Actual Down Time	Efficiency of Engineer	Not shown	
Standard Rerun Percentage (data) / Actual Rerun Percentage	Effectiveness of Input Preparation or Control	Not shown	
Standard Rerun Percentage (program) / Actual Rerun Percentage	Programmer Testing Quality	Not shown	
Standard Training Time / Actual Training Time	Effectiveness of Trainers	Not shown	

These, and others, can be easily constructed from a variance report. Data obtained in this manner should not be taken at face value; investigation is required in almost all cases to determine causes not directly reflected in the statistics. An increase in set-up time, causing a decrease in operator efficiency, may be caused by a new operator, the temporary replacement of an operator, or the prolonged absence of the supervisor. Whatever the cause, management will learn a great deal about the effects of these conditions on the total cost of operation, so that preventive action can be taken to avoid recurrences where the cost is seriously increased. Measures of programming quality are explained in Chapter IX. It may be well worthwhile to incur a limited number of program errors if the quantity produced is consistently above the standard; conversely, a significant decrease in the productivity of the program testing may be incurred if perfect quality is demanded.

Cost, Rental, and Service Accounting

Most installations service departments other than their own. Accordingly, it becomes necessary to develop a charging system, to properly allocate the costs of operation in proportion to the usage or benefits derived. Since many installations include satellite computers or other

peripheral equipment, the development of a cost accounting system requires a considerable amount of analysis. It is also necessary to produce a report showing total chargeable and non-chargeable time, in order to determine the total rental due the manufacturer.

Cost-Accounting Methods.—Five methods are discussed below by which using departments can be assessed for their part of the costs, the costs of production, overhead, rerun time, and so on, as well as the costs of amortizing the development program.

Method 1—Straight Overhead (Indirect Costing).—The simplest method is to charge the entire cost of computer center operation and development to general overhead or general research and development expense. Since these expense accounts are usually allocated to operating departments according to a predetermined percentage, this method has the effect of loading the largest department with the largest charge for computer operations, even if it is not the largest computer user.

Method 2—Straight Allocation.—The second method, in order of simplicity, is to charge using departments according to a fixed percentage of computer use. This percentage is usually determined at the beginning of each year, based on an historical average or on an intuitive analysis made by the department manager. It is better than Method 1, since it applies the charge in some proportion to use, but changes in use ratios are not reflected if they occur in the middle of the year. The total costs are the sum of

- Machine rental
- Salaries
- Overhead charges
- Supplies and miscellaneous
- Amortization of development cost (a 5-year charge-off is possible).

The total charges are then distributed, according to the predetermined percentages, to each department, including the data processing department or the computer laboratory.

Method 3—By Use of Main Computer Time.—The third method, which is very common, is the distribution of all departmental costs according to the use of main computer time. All of the operating costs of the department are summarized, as before, and divided by the total number of productive hours that can be directly charged to a specific user. This provides a direct hourly rate of computer operation for determining charges. If it also is possible to allocate directly set-up, testing and compiling, a more realistic figure may be obtained. The total costs of operation divided by the number of total hours chargeable to a bona fide user will give the correct allocation:

Thus, the total charge to a user is

$$\text{Number of hours used} \times \frac{\text{Total operating costs}}{\text{Total hours chargeable}}$$

Figure 8-8 illustrates a report used to make this determination by user. The installation has a 7090 and a 1401, but the total cost distribution is by 7090 hourly use only, based on the assumption that one hour of 7090 time generates 1 hour of 1401 time, and all associated costs are included. Figures 8-6 to 8-11 demonstrate the complete accounting system required to back-up the charges as follows:

Figure 8-6 is the basic log record produced in chronological sequence.

Figure 8-7 shows the same information sorted by account (user) within job number (program)—to establish specific use charges.

Figure 8-8 is a summary by account, used to make the distribution.

IBM 7090 UTILIZATION LOG

JUN 01 1962

TIME	CODE	JOB	SEC	W.O.-EWA	CUSTOMER	ACCOUNT	PERSONNEL	SYS	CH TP R/P	HOURS
00 00 0	01	2362	008	31 41431501	BURNES	7221 01	FLACK 4		2 06 0	.127
7 4	10								2 11 3	.025
00 09 1	04							M	2 08 2	.006
00 09 3	01	3040	001	31 41431501	BURNS	7221 01	FLACK 1	M	2 06 0	.058
12 6	10								2 11 3	.017
00 14 0	04							M	2 08 2	.005
00 14 2	01	3040	001	31 41431501	BURNS	7221 01	FLACK 4	M	2 06 0	.025
15 5	10								2 11 3	.014
00 16 4	04							M	2 08 2	.003
00 16 5	01	3094	001	51 1007	PARR	4411 01	M MACHINE1	M	1 02 0	.003
00 17 0	01	1177	010	51 1003	PARR	4411 01	M MACHINE	M	2 05 0	.069
00 21 1	04							M	2 08 2	.003
00 21 2	01	3032	000	51 1003	PARR	4411 01	M MACHINE	M	2 08 0	.313
40	10								2 11 3	.020
00 41 2	10							M	2 11 3	.131
00 49 1	04							M	2 08 2	.003
00 49 2	03	3225	000	51 1003	SMITH	4411 01	HARVEY 1	M	2 07 2	.003
00 49 3	03	3225	000	51 1003	SMITH	4411 01	HARVEY 1	M	2 07 2	.005
00 49 5	03	3225	000	51 1003	SMITH	4411 01	HARVEY 1	M	2 07 2	.003
00 50 0	03	3225	000	51 1003	SMITH	4411 01	HARVEY 1	M	2 07 2	.003
00 50 1	03	3225	000	51 1003	SMITH	4411 01	HARVEY 1	M	2 07 2	.002
00 50 2	03	3225	000	51 1003	SMITH	4411 01	HARVEY 1	M	2 07 2	.003
00 50 3	03	3225	000	51 1003	SMITH	4411 01	HARVEY 1	M	2 07 2	.003
00 50 4	03	3225	000	51 1003	SMITH	4411 01	HARVEY 1	M	2 07 2	.011
00 51 2	03	3225	000	51 1003	SMITH	4411 01	HARVEY 1	M	2 07 2	.064
00 55 1	03	3225	000	51 1003	SMITH	4411 01	HARVEY 1	M	2 07 2	.075
00 59 4	10							M	2 11 3	.011
01 00 2	03	3078	004	51 19726200	SANFORD	7244 01	HENDERLIGHT1	M	2 07 2	.008
01 00 5	04							M	2 08 2	.009
01 01 2	02	3078	004	51 19726200	SANFORD	7244 01	HENDERLIGHT2	M	2 10 0	.022
01 02 4	10								2 11 3	.028
01 04 2	04							M	2 08 2	.005
01 04 4	02	3171	000	51 1007	SMITH	4411 01	WHITE 8	M	2 10	.011
1 5 2	10								2 11 3	.037
01 07 3	04							M	2 08 2	.002
01 07 4	02	3085	000	31 41730000	NUCLEAR	7612 01	DOUGLASS 1	M	1 02 0	.006
01 08 0	04							M	2 08 2	.003
01 08 1	02	3074	001	41 51049021	WILLIAMS	7221 05	MCNEILL1	M	2 02 0	.025
01 09 4	03	3204	000	21 72522705	ANDRUS	7222 05	BASKIN1	M	2 07 2	.008
01 10 1	03	3204	000	21 72522705	ANDRUS	7222 05	BASKIN12	M	2 07 2	.006
01 10 3	03	3080	000	31 41651001	CARPEROS	7222 01	BASKIN	M	2 07 2	.008
01 11 0	10							M	2 11 3	.219
01 24 1	01	1558	000	21 21020000	HARRIS	7221 01	NORSWORTHY		2 06 3	.184
01 35 1	10						SET-UP		2 11 3	.055
01 38 3	01	1305	000	21 70532004	LEE	7694 01	NORSWORTHY		2 07 3	.084
01 43 3	10						SET-UP		2 11 3	.025
01 45 0	15	1305	000	21 70532004	LEE	7694 01	NORSWORTHY		2 02 0	.200
01 57 0	02	3024	000	21 2102	REPROD.TAPES	7221 07	WADLE		1 02 1	.166
02 07 0	13								2 11 3	.767
02 53 0	02	3024	000	21 2102	SAYER 1916	7221 07	WADLE		2 12 3	.100
02 59 0	02	3024	000	21 2102	SAYER 1	7221 07	WADLE		2 12 3	.150
03 08 0	10								2 11 3	.167
03 18 0	02	3024	000	21 21020000	WADLE 6LRW	7221 07	WADLE		2 12 3	.350
03 39 0	10								2 11 3	.183
03 50 0	02	3024	000	21 21020000	WADLE 6LRM	7221 07	WADLE		2 12 3	.133

Fig. 8-6. Equipment Utilization Log. (*Courtesy*, Lockheed-Georgia Company, A Division of Lockheed Aircraft Corporation)

2. SUMMARY BY JOB / ACCOUNT
01 JUNE 1962

JOB	CODE	PROGRAMMER	CUSTOMER	ACCOUNT	W.O.-EWA	TIME OF DAY	THIS DAY HOURS	THIS DAY PERCENT	THIS A.P. (050762) HOURS	THIS A.P. (050762) PERCENT	THIS YEAR HOURS	THIS YEAR PERCENT
1177.010	01	M MACHINE	PARR	4411-01	1003-0000	0017	.07					
1177.010	01	M MACHINE	PARR	4411-01	1003-0000	0636	.06					
1177.010	01	M MACHINE	PARR	4411-01	1003-0000	1145	.05					
1177.010	01	M MACHINE	PARR	4411-01	1003-0000	2152	.46					
1177.010 GELAC NUMERICAL CONTROL SYSTEM					TOOL DESIGN DEPARTMENT		.64	4.05	6.89	2.76	41.44	2.72
1305.000	01	NORSWORTHY	LEE	7694-01	7053-2004	0138	.08					
1305.000	01	NORSWORTHY	LEE	7694-01	7053-2004	2056	.58					
1305.000 NUCLEAR					PHYS ANAL AND INSTRU. DEPT.		.66	4.18	9.80	3.93	15.45	1.01
1488.004	02	EUBANK 4	REED	7244-01	1972-6200	0552	.65					
1488.004	02	EUBANK 7	REED	7244-01	1972-6200	0804	.02					
1488.004	02	EUBANK 7	REED	7244-01	1972-6200	0822	.44					
1488.004	02	EUBANK 2	SANFORD	7244-01	4143-1501	0853	.57					
1488.004	02	EUBANK 9	SANFORD	7244-01	1972-6200	0929	.01					
1488.004	02	EUBANK 10	REED	7244-01	1972-6200	1131	.10					
1488.004	02	EUBANK 4	REED	7244-01	1972-6200	1616	.66					
1488.004	02	EUBANK 2	SANFORD	7244-01	4143-1501	1700	.24					
1488.004	02	EUBANK 10	REED	7244-01	1972-6200	1714	.79					
1488.004	02	EUBANK 9	SANFORD	7244-01	1972-6200	1802	.01					
1488.004	02	EUBANK 2	SANFORD	7244-01	4143-1501	1945	.59					
1488.004 EXECUTIVE PROGRAM FOR RANDOM DATA ANALYSIS					ENG. FLIGHT TEST DIV.		4.08	25.64	24.31	9.73	92.37	6.05
1488.026	02	ADAMS	SANFORD	7244-01	4143-1502	0810	.00					
1488.026	02	ADAMS2	SANFORD	7244-01	4143-1502	1126	.04					
1488.026	03	ADAMS	SANFORD	7244-01	4143-1502	1506	.00					
1488.026	02	ADAMS	SANFORD	7244-01	4143-1502	1507	.06					
1488.026	02	ADAMS2	SANFORD	7244-01	4143-1502	1826	.05					
1488.026	02	ADAMS 1	SANFORD	7244-01	4143-1502	1900	.04					
1488.026 FILTER REVISION TO 1488.004					ENG. FLIGHT TEST DIV.		.20	1.26	1.51	.60	1.51	.10
1527.000	02	SOUTHWELL	CRENSHAW	7221-05	2102-0000	1149	.02					
1527.000	02	SOUTHWELL	CRENSHAW	7221-05	2102-0000	1924	.30					
1527.000 SWEPT SURFACE FLUTTER ANALYSIS					DYNAMICS GROUP		.32	1.99	.56	.22	1.28	.08
1558.000	01	NORSWORTHY	HARRIS	7221-01	2102-0000	0124	.18					
1558.000	01	NORSWORTHY	HARRIS	7221-01	2102-0000	2228	.16					
1558.000 NA FRAME PROGRAM					STRENGTH ANALYSIS DEPT.		.34	2.17	5.86	2.35	33.10	2.17
1569.002	01	HARRIS-1	HYCHE	7244-01	4165-1004	2045	.01					
1569.002 REVISION TO T.O.L. CAMERA PROGRAM					ENG. FLIGHT TEST DIV.		.01	.06	.13	.05	.16	.01
1603.000	01	NIX	BARTNICK	7221-05	4143-1501	1943	.00					
1603.000 LEAST SQUARE FITS TO GROUPS OF DATA					DYNAMICS GROUP		.00	.03	.09	.04	.15	.01
1607.000	02	HENDERLIGHT	SANFORD	7244-01	1972-6200	0951	.24					
1607.000 PROCESSING OF FLIGHT TEST MAGNETIC TAPES					ENG. FLIGHT TEST DIV.		.24	1.49	.24	.09	.24	.02
2108.000	02	HARRELL 1	CARPEROS	7222-01	1972-6200	1433	.00					
2108.000	02	HARRELL 2	CARPEROS	7222-01	1972-6200	1433	.03					
2108.000 NORMAL LANDING					DIGITAL PROGRAMMING GROUP 1		.03	.21	.70	.28	.79	.05
2196.000	02	ALFORD2	CARPEROS	7222-01	1972-6200	0811	.04					

Fig. 8-7. Usage Analysis by Job and Account. (*Courtesy*, Lockheed-Georgia Company, A Division of Lockheed Aircraft Corporation)

3. SUMMARY BY ACCOUNT
01 JUNE 1962

ACCOUNT	ACCOUNT TITLE	THIS DAY		THIS A.P. (050762)		THIS YEAR	
		HOURS	PERCENT	HOURS	PERCENT	HOURS	PERCENT
4411-01	TOOL DESIGN DEPARTMENT	1.84	11.55	23.47	9.40	117.06	7.67
7207-01	AERODYNAMICS DEPARTMENT	.23	1.45	4.29	1.72	61.00	4.00
7207-02	THERMO AND PROPUL. DEPT.	.63	3.96	3.01	1.21	15.19	1.00
7208-02	OPERATIONS EVAL DEPT.	.25	1.55	1.78	.71	19.16	1.26
7212-01	ENGINEERING LOFT GROUP	.00	.02	.64	.26	3.79	.25
7221-01	STRENGTH ANALYSIS DEPT.	.89	5.61	30.64	12.27	234.60	15.38
7221-02	WEIGHT ANAL AND CONT DEPT.	.22	1.38	1.54	.62	2.33	.15
7221-03	C-130 AND GEN. LOADS GROUP	.01	.06	2.13	.85	20.23	1.33
7221-05	DYNAMICS GROUP	.78	4.89	35.79	14.33	96.69	6.34
.221-06	FLUTTER GROUP	.11	.72	10.25	4.10	75.33	4.94
7221-07	STRUC ANAL METH AND PRO DPT	4.05	25.48	58.35	23.37	280.10	18.36
7221-09	C141 LOADS GROUP	.69	4.33	4.66	1.87	95.60	6.27
7222-01	DIGITAL PROGRAMMING GROUP 1	.44	2.79	2.85	1.14	17.07	1.12
7222-02	DIGITAL PROGRAMMING GROUP 2	.01	.06	1.30	.52	16.59	1.09
7222-03	OPERATIONS GROUP	.15	.94	3.39	1.36	28.00	1.84
7222-05	ANALYSIS GROUP	.09	.54	1.24	.50	7.11	.47
7244-01	ENG. FLIGHT TEST DIV.	4.74	29.80	42.37	16.97	192.27	12.60
7612-01	NUCLEAR ANALYSIS DEPT.	.06	.39	2.35	.94	91.96	6.03
7620-01	NUCLEAR LAB DIVISION	.04	.26	.98	.39	44.49	2.92
7694-01	PHYS ANAL AND INSTRU. DEPT.	.66	4.18	9.80	3.93	38.21	2.50
9401-01	MASTER SCHEDULING	.01	.05	1.69	.68	14.34	.94

Fig. 8-8. Usage Summary by Account. (*Courtesy*, Lockheed-Georgia Company, A Division of Lockheed Aircraft Corporation)

Figure 8-9 summarizes the charges by work order (EWA), a breakdown within the account which the user needs to charge the appropriate project for the expense.

Figure 8-10 is the breakdown by computer use category, divided into "hours used," which are chargeable, and "hours not used," which are not chargeable.

Figure 8-11 is the same report by category for all of the peripheral equipment, whose overtime charges are based on a different rate structure.

Method 4—By Print Volume.—In a commercial installation, the use of main frame time does not always determine the total charges effectively. The reason for this is that some users require more volume of output data than others who are more interested in record maintenance. An inventory system, for example, produces output only periodically, often in summary form. By contrast, an engineering release system may produce thousands of blueprint change orders daily—one major part change may cause a "Christmas tree" of sub-assembly and sub-part changes. As a result, some installations have found a more representative charge base in the total print volume produced. This method requires the programs or hardware to count printer output lines and allocate these to the various accounts. Total operating cost is divided by total chargeable printer lines (in thousands) to arrive at a "unit" cost. The user is then charged in accordance with his use of the printer. The total charge is:

$$\text{No. of chargeable print lines} \times \frac{\text{Total operating cost}}{\text{Total chargeable printer lines}}$$

WORK ORDER - EWA	THIS DAY		THIS A.P. (050762)		THIS YEAR	
	HOURS	PERCENT	HOURS	PERCENT	HOURS	PERCENT
0000-0000	.00	.00	.05	.02	.05	.00
1002-0000	.00	.00	.28	.11	6.02	.39
1003-0000	1.44	9.04	16.41	6.57	81.89	5.37
1007-0000	.40	2.52	8.08	3.23	47.09	3.09
1008-0000	.01	.05	1.69	.68	14.19	.93
1101-0000	.00	.00	.04	.02	3.24	.21
1102-6501	.00	.00	.00	.00	.72	.05
1106-0000	.00	.00	.00	.00	.54	.04
1202-0005	.00	.00	.00	.00	7.81	.51
1454-0000	.25	1.55	2.09	.84	8.84	.58
1475-0000	.00	.00	.00	.00	.37	.02
1491-0000	.00	.00	.03	.01	1.37	.09
1693-0000	.00	.00	.00	.00	.03	.00
1701-0000	.00	.00	.00	.00	2.19	.14
1972-6200	3.24	20.40	26.19	10.49	118.27	7.75
2101-0000	.00	.00	.75	.30	.75	.05
2102-0000	6.15	38.67	124.86	50.00	765.88	50.20
2102-7004	.00	.00	.00	.00	.01	.00
2120-0000	.00	.00	.00	.00	.05	.00
2148-1001	.00	.00	.00	.00	19.16	1.26
2148-3001	.00	.00	.79	.32	3.98	.26
2657-0000	.00	.00	.00	.00	.10	.01
3001-0000	.00	.00	.00	.00	.36	.02
3006-0000	.00	.00	.00	.00	.28	.02
3037-0001	.00	.00	.00	.00	.20	.01
4001-0000	.00	.00	.04	.01	.04	.00
4127-0000	.00	.00	.00	.00	.02	.00
4143-1501	1.99	12.51	22.00	8.81	49.37	3.24
4143-1502	.20	1.26	1.51	.60	2.37	.16
4143-1504	.00	.00	.00	.00	1.30	.09
4143-1506	.00	.00	.10	.04	2.12	.14
4143-1508	.00	.00	.06	.02	.06	.00
4162-0000	.00	.00	.12	.05	4.22	.28
4162-1610	.00	.00	.01	.00	.29	.02
4165-1001	.32	2.00	11.88	4.76	75.08	4.92
4165-1004	.03	.21	2.08	.83	8.13	.53
4168-0000	.00	.00	.00	.00	.07	.00
4170-0000	.00	.00	.29	.12	14.08	.92
4172-0000	.00	.00	.01	.00	.59	.04
4173-0000	.06	.39	1.93	.77	10.38	.68
4176-0000	.00	.00	.00	.00	56.20	3.68
4197-9022	.00	.00	.00	.00	.00	.00
4198-4002	.00	.00	.00	.00	.17	.01
4201-0000	.00	.00	.00	.00	.00	.00
4201-4002	.00	.00	.00	.00	.42	.03
4203-4006	.00	.00	.18	.07	.18	.01
4204-4001	.00	.00	.65	.26	.65	.04
4210-0000	.00	.00	.31	.12	.31	.02
4404-0000	.10	.65	.26	.10	.26	.02
4427-0000	.00	.00	.29	.12	1.58	.10
4487-1003	.00	.00	.00	.00	3.08	.20
4487-2001	.00	.00	2.58	1.03	8.86	.58
4487-5010	.00	.00	.00	.00	.51	.03

Fig. 8-9. Usage Summary by Work Order. (*Courtesy*, Lockheed-Georgia Company, A Division of Lockheed Aircraft Corporation)

Method 5—Direct Equipment Charging.—The most exact method, though the most costly, is to charge directly for each unit of equipment used. This generally involves the use of a clocking device on each operating unit, with a separate log for each. As a result, the total equipment time of all components, satellites and peripheral gear is maintained. An hourly rate is then determined for each unit, as follows:

a. Summarize the total operating cost of the department
b. Subtract the total equipment rental cost

IBM 7090 UTILIZATION SUMMARY

APRIL 1962

1. SUMMARY BY CODE

	THIS A.P. (010162) HOURS	PERCENT	THIS YEAR HOURS	PERCENT
01 PRODUCTION	205.48	24.46	723.52	23.93
02 PROGRAM CHECKOUT	107.96	12.85	408.57	13.51
03 ASSEMBLY - COMPILATION	32.95	3.92	143.84	4.76
04 EXECUTIVE PROGRAM OPERATION	14.15	1.68	58.52	1.94
05 OPERATOR ERROR	4.62	.55	11.59	.38
06 DATA PREPARATION ERROR	1.78	.21	2.67	.09
07 COMPUTER ERROR	6.75	.80	18.03	.60
08 TAPE ERROR	3.23	.38	8.10	.27
TOTAL HOURS USED	376.92	44.87	1374.84	45.46
10 SET-UP	111.39	13.26	394.36	13.04
11 IDLE	117.70	14.01	368.08	12.17
12 SCHEDULED MAINTENANCE	38.30	4.56	130.79	4.32
13 UNSCHEDULED MAINTENANCE	7.04	.84	39.13	1.29
14 UTILITIES FAILURE	.69	.08	7.12	.24
15 NECESSARY TESTING	22.07	2.63	61.65	2.04
16 NOT SCHEDULED	165.91	19.75	648.11	21.43
TOTAL HOURS NOT USED	463.10	55.13	1649.24	54.54
TOTAL HOURS	840.02	100.00	3024.09	100.00

Fig. 8-10. Usage Summary by Category. (*Courtesy*, Lockheed-Georgia Company, A Division of Lockheed Aircraft Corporation)

c. Divide the remainder by total equipment rental, to determine the unit overhead charge

d. The unit overhead charge is

$$\frac{\text{Total cost - Equipment rental}}{\text{Equipment rental}}$$

e. For each machine, compute the hourly charge by dividing its total rental by the total chargeable hours

f. Multiply the hourly charge by 1 + the overhead factor, to establish a total hourly charge for the equipment item

For example, assume that the total cost is $25,500 in April, of which $11,000 is equipment rental. The computer rental is $5,500, with 190

IBM 7090 MONTHLY EQUIPMENT USAGE

APRIL 1962

CODE	7090	729	711	716	7607B
01	205.477	1574.181	110.787	107.889	182.597
02	107.961	712.285	29.873	33.626	82.873
03	32.951	230.837	.000	34.191	32.951
04	14.154	113.232	.000	14.154	14.154
05	4.616	28.078	1.767	1.887	4.338
06	1.777	14.413	1.280	.983	1.772
TOTAL (01-06)	366.936	2673.026	143.707	192.730	318.685
07	6.751	54.809	3.047	5.700	6.367
08	3.233	17.186	.283	1.549	2.192
TOTAL (07-08)	9.984	71.995	3.330	7.249	8.559
09	.000	.000	.000	.000	.000
10	111.388	1225.331	111.397	111.397	111.388
11	117.703	1294.733	117.703	117.703	117.703
12	38.304	421.344	38.304	38.304	38.304
13	7.037	77.314	7.006	7.037	7.037
14	.686	7.546	.686	.686	.686
15	22.071	132.233	8.749	10.052	21.253
16	165.907	1824.977	165.907	165.907	165.907

Fig. 8-11. Usage Summary by Equipment and Category. (*Courtesy*, Lockheed-Georgia Company, A Division of Lockheed Aircraft Corporation)

hours chargeable to a user; an off-line unit rents for $220 with 156 hours accountable.

The overhead charge per dollar of rental is: $\frac{\$25{,}500 - \$11{,}000}{11{,}000} = \$1.32$

The computer hourly charge is $\frac{\$5{,}500}{190} = \$28.95$

The off-line unit's hourly charge is $\frac{\$220}{156} = \$1.41$

The accounting charge for the computer is $\$28.95 \times (\$1.32 + 1) = \$67.16$ per hour

The accounting charge for the off-line unit is $\$1.41 \times (\$1.32 + 1) = \$3.28$ per hour

On the basis of the log, the hourly usage of each item is multiplied by its rate and charged to the appropriate account.

In all five charging systems, rerun time, unscheduled maintenance, and other categories of this nature should not be charged to the account

on whose time they occur, but should be included in the overhead category.

Determination of Trends, and Their Impact

Because of the length and nature of the development program, long delivery schedules, and rapid technological obsolescence, management must today plan the data processing requirements of the next three years. It is therefore important to plot critical figures on a monthly graph, to show trends up or down in equipment utilization. Equipment use is affected by many variables; for example, business trends, maturity of the computer installation, and, in scientific equipment usage, engineering employment. When a new piece of equipment, or a new system is introduced there is a tendency to experiment: engineering usage increases sharply in the first months after delivery. In commercial data processing executives demand greater utilization, after the first equipment rental bill is submitted.

Graphic representation of experience by type of use helps in determination of the elements forcing change, or a general trend. If, for example, set-up time becomes excessive, a monitor may be introduced. If productive time increases, and a large part of this is attributable to data sorting, a separate smaller machine may be ordered for sorting alone. Statistics should similarly be kept for printing loads, input conversion loads, and test time, keeping in mind the approximate maximums for which the current system makes provision.

Figure 8-12 is an approximate graph by quarterly periods of the equipment utilization of a small manufacturing company. From the graph, it can be seen that the introduction of a new monitor system markedly reduced the widening gap between productive time and other time (Last Quarter, 1961). A projection of the trend shows, that with a continued growth rate of the same slope, the current equipment will be fully utilized (three shifts) in the last quarter of 1965. Because of the complexity of the required programming, management will therefore have to initiate a conversion program to a larger capacity system before the end of 1963, assuming a 2-year lead time.

Guidance of Management Action

The final objective of a detailed reporting system built around established equipment performance standards is to suggest and guide appropriate management action to correct an unsatisfactory situation or to improve effectiveness of control. Typical actions that result from analysis of standards variance are indicated below:

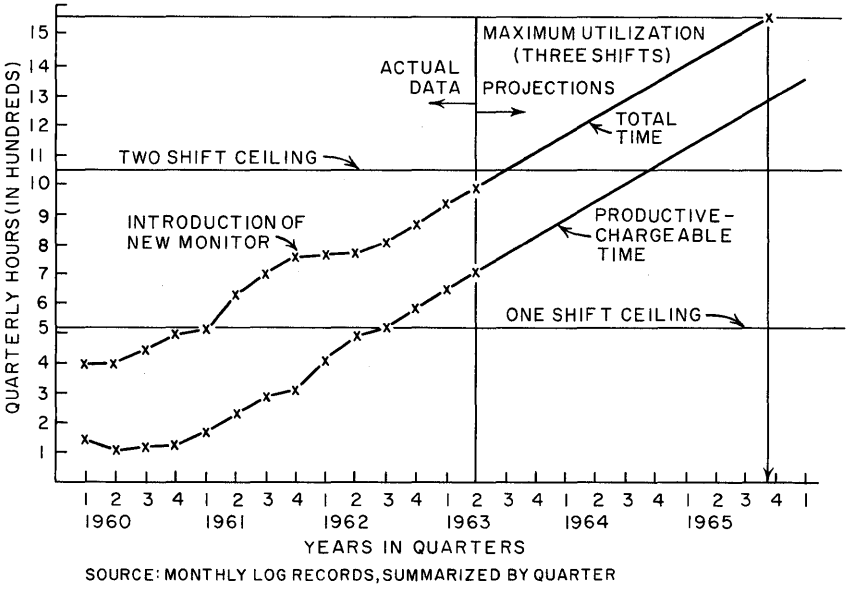


Fig. 8-12. Example of Historical and Projected Computer Utilization.

Change of Personnel.—The reporting system provides a record of performance in time; variances may be caused by the presence or absence of certain staff members, in supervisory or non-supervisory positions. In the case illustrated, the rerun percentage caused by data error suffered greatly when the keypunch supervisor was absent. In other situations, the absence of a poor supervisor may increase performance of his staff. Management will be able to relate such fluctuations in performance and determine if personnel change is warranted.

Installation of Incentives.—By comparing performance against the standards of other organizations, it is possible to determine if the overall performance is better or worse than others. It may be worse because of personnel quality, and often because of poor morale. Incentives can be provided by management to build morale and to increase the competitive spirit among the staff.

Change of Layout.—An unsatisfactory computer room layout will be reflected in a high percentage of set-up time or a high percentage of manual handling.

Change of Procedure.—Delays in delivery of input to the computer center and delays in the processing of output reduce adherence to schedule and increase the amount of handling required.

Modification of Standard.—If the variance is consistently large in one direction, and no reasonable explanation exists, it is possible that the standard is wrong. Management should modify the standard when it is

fully satisfied that there is no other external cause for the consistent variance.

Measures of Quality in Equipment Performance

Pertinent ratios were presented earlier for determining the effectiveness of operating personnel. These indicated the efficiency of set-up or equipment maintenance on a quantitative basis. If quantity alone is emphasized it may improve at the expense of quality. Set-up time, for example, can be reduced greatly by pressure upon operators, but the registration of preprinted forms to the information may no longer be exact and the operator error ratio may gradually increase.

Qualitative measures must therefore be established alongside of quantitative measures. These have been already established in part as operating methods standards. Certain additional basic quality indications should be kept under management surveillance such as:

- Printer line-up and registration: An operator in a hurry will often sacrifice exact registration. This can easily be checked and corrected through proper supervision.
- External tape labeling: The completeness, accuracy, and neatness of the external label on all tape files can be checked quickly, and omissions or errors traced.
- Log record maintenance: The same factors can be evaluated in the manner in which the log record is maintained.
- General housekeeping: Fast moving operators will generally avoid good housekeeping, since it is time consuming. Replacement of excess paper and cards and general clean up of the console and computer area are often neglected in order to improve operating performance.

Measures of quality and quality control procedures will vary from one installation to another; management's awareness of the need to control quality while improving performance is all that is really necessary.

CHAPTER SUMMARY

Chapter VIII is the first of two chapters dealing with establishment and use of performance standards. It introduces the general approach to performance evaluation, consisting of

Standards development
Scheduling and
Analysis of performance data

Chapter VIII deals primarily with the establishment of standards for equipment, although it involves an evaluation of operating personnel. Equipment standards are derived for each possible category of utilization, and they are summarized in a Table, on pages 234-235. The reader is also introduced to the concept of performance parameters-variables which affect the performance of equipment or personnel in a manner which can be determined, either empirically or through evaluation and analysis. Thus, the complexity of a program affects the number of test shots which are required, and the number of input or output units of a program affect its set-up time.

The second step in an equipment performance standards program is the development of a performance schedule, followed by the detailed analysis of data. The analysis has as its main objectives the detection and correction of variances, the accounting for the time used and the evaluation of personnel effectiveness. The latter should be accomplished only if appropriate measures of quality are established and enforced along with the necessary measures of quantity.

Questions for Review

1. Develop a table of productive time standards for the following application:
   Inputs: Master file—50,000 items 2 $\times$ 300
   Table tape—1000 items 1 $\times$ 100 read once for each detail item
   Detail tape—10,000 items 1 $\times$ 80
   Outputs: Updated master file
   Trial balance report—all detail items + 500 control totals
   Report tape—10,000 items, 1 $\times$ 180
   Single channel system with process overlap.
2. What is the "critical volume"?
3. Graph the results.
4. Repeat problem 1 for a two-channel system.
5. Develop a formula for set-up time for this run.
6. Indicate the categories of utilization which apply to your installation.
7. Develop a methodology for standards for each of the above in Question 6.
8. Develop a layout of a reporting system to evaluate the results.
9. Develop the critical ratios for evaluation of the operating staff.
10. Develop the quality measures to go along with the evaluation of the ratios indicated in Question 9.

Chapter IX

PERFORMANCE STANDARDS: PERSONNEL

INTRODUCTION

Chapter VIII was concerned with the development and use of performance standards for equipment and operating personnel. Most of this Chapter is concerned with performance measurement of programming personnel and a methodology is advanced for the development and use of performance standards for programming tasks. Since programming is a more exact and better defined art than systems analysis, most previous experience has been with programming standards. However, this Chapter shows the same methods applied to systems analysis.

The program parameters defined in Chapter VIII—size, complexity and input-output complexity—are also used to measure programming tasks. Similar parameters are shown for systems analysis, but these are untried.

The formulas and relationships given in this Chapter have been used by the author in a number of installations. In each case, their accuracy has been verified by continued use. They should not, however, be used by the reader without consideration of his installation's particular procedures and problems. The formulas given here demonstrate an effective method and give the reader a starting point. In use, modification should be made for different programming methods or problems.

STANDARDS FOR PROGRAMMING PERSONNEL

The methods of establishing and using standards for programming personnel are similar to those used in Chapter VIII. The major steps are:

- Listing of the tasks to be performed
- Grouping of these tasks into major sets
- Development of standard relationships between these tasks and the time required to perform them

- Development of a schedule
- Gathering of data
- Evaluation of the data
- Management action
- Establishment of quality measures

Listing of the Tasks

The basic tasks to be performed in writing a program are listed in the sequence defined by the established methods standards (as described in Chapters IV and V):

- Read the job specification manual
- Review the program functions
- Analyze the layouts provided
- Review the program flowchart
- Develop a macro-block diagram
- Assign block letters to distinct segments
- Develop micro-block diagrams for each of the segments
- Review the macro- and micro-block diagrams
- Translate the program logic into symbolic language
- Develop coding for the item layouts
- Add the necessary standard subroutines
- Desk check the translation
- After key-punching and necessary EAM checking, validate the preliminary listing
- Prepare the required test data
- Assemble the program
- Test the program
- Perform a production test with data supplied by the analyst
- Assist in performance of a systems test
- Prepare the program documentation
- Assist in conversion
- Update the block diagrams to include all corrections
- Turn the program over to operations

Some of these tasks take a large amount of time, like micro-block diagramming. Others are completed in a matter of hours, such as the addition of standard subroutines. The accurate development of performance standards requires grouping of these tasks into measurable elements.

Grouping of Tasks

Three task groupings are shown below. One has been found realistic for a new development program. The second set, used in establishing standards for converting programs to another machine, and the third set, used for maintenance programming, are modifications of the first.

New Development Groups

Group 1—Macro-Logic:

Job specification analysis
Review of functions
Layout analysis
Program flowchart
Macro-block diagram

Group 2—Micro-Logic:

Micro-block diagramming
Logic review

Group 3—Coding:

Translation
Item layout coding
Standard subroutines

Group 4—Desk Checking:

Desk Checking
Listing validation
Test data preparation

Group 5 Testing:

Assembly or compilation
Program testing
Production testing
Systems testing
Conversion and installation assistance

Group 6—Documentation:

Documentation
Proofreading
Block diagram updating
Program turnover

Conversion Program Task Grouping

If the conversion is to be made to a completely incompatible machine and the existing documentation is unavailable or outdated, the basic grouping is similar. The only possible difference is in test data preparation. Presumably a sufficient number of test cases would be available in an already existing installation, and the desk checking task would be reduced and combined with the 6th task—documentation. For compatible machines or compatible languages the only tasks which are required are review, assembly and production testing.

Maintenance Programming

If documentation is current, the maintenance task groupings are

Group 1—Review of existing documentation
Review of change specifications

Group 2—Development of change logic

Group 3—Coding and desk checking of the change

Group 4—Test and assembly

Group 5—Documentation updating
Installation and turnover

In general, separate standards and rating parameters will be developed for new development programming and for maintenance programming, but not for conversion programming since in most cases the system is redesigned when the new machine system is ordered.

Relationship of Tasks and Time

The next step in the development of programming standards is establishment of meaningful time relationships between the programming tasks, and the nature of the program. This is a most critical function; it requires the application of a considerable amount of judgment and experience.

Task Group 1—Macro-Logic

The initial approximation of time can be done as follows:

Reading the job specification	½ to 1 day
Review or listing of the functions	¼ to ½ day
Analysis of layouts	¼ day
Macro-block diagram, program segmentation	½ to 4 days (depends on complexity)

The first two items are also dependent on the size of the program. The total time lies between ½ day and 6 days. The difference is a function of the program complexity, and less of program size.

Task Group 2—Micro Logic

Block diagram development:	Approximately ¼ day for each block for a simple program; up to 10 days, for a complex program.
Logic review:	From ½ day to 2 days

The micro-diagramming function depends on two factors. It shows:

a linear relationship with program size
a geometric progression dependent on program complexity.

Task Group 3—Coding

When block diagramming is done well, coding effort is almost directly proportional to program size. Somewhere between ½ and 1½ days are needed to complete 10 pages (one unit size) if the program is simple. A complex program may require from 2 to 4 additional days to insure that the linkage between blocks is properly established.

Task Group 4—Desk Checking

Program checking is a function of both size and complexity. Test data preparation is largely a function of size. The total time for both tasks usually varies between 2 and 7 days.

Task Group 5—Testing

The testing function includes compilation, which requires 2 to 4

man-hours of programmer time for review and error correction. The program time for each test shot depends on testing practices as follows:

5 hours if testing is done at a remote computer by the programmer
4 hours if testing is done at the plant location by the programmer
2½ hours if testing is done by the operators following a documented test plan

The number of test shots is, of course, a function of program complexity and size and, in some cases, of the input-output complexity.

Task Group 6—Documentation

The time required to document a program is that necessary to produce the

- General description
- Detailed description
- Operator's instructions
- Miscellaneous sections of the manual

The bulk of the manual will already have been completed when the block diagrams and flowcharts are up-to-date. The standard time for the above listed functions is a function of size and complexity. The number of pages needed is estimated first and the time is estimated from this as shown.

Typical Formulas for Development Programming

The formulas given below have been developed for the IBM 1401, 7080, 7090, and the UNIVAC 490® machine systems. Although some tasks appear to be machine-independent, such as macro-logic and micro-logic, there are differences among machine systems because differences in memory size force differences in systems design. A system designed for an 8,000 character 1401 would contain smaller and less complex programs than it would for a 160,000 position 7080, and a program rated C-5 for a 1401 would probably be a much smaller and less complex program than a similarly rated program for the 7080. Differences between the systems also arise from the number and type of available computer commands. The 1401 has only 40 basic instructions while the 7090 has 160 and the 7080 has 64.

The formulas have been tested under the specific conditions stated. They are intended to serve as guides in establishing formulas for similar, typical installations.

Standards for the IBM 1401 Tape System

Memory Positions: 8,000 characters
Coding: SPS III
Average Program: C-5/4 *; documentation standard
Unit of Size: 10 pages of coding

Task 1—Macro-Logic

Man-days for level of complexity	
½	A
1	B
1	C
1½	D
2½	E

Task 2—Micro-Logic

	Man-days for level of complexity	
	0	A
	2	B
½ day per unit of size *plus*	4	C
	6	D
	8	E

Task 3—Coding in SPS III

	Man-days for level of complexity	
	0	A
	1	B
1 day per unit of size *plus*	2	C
	3	D
	4	E

Task 4—Desk Checking

Man-days	*Complexity and Size*
2½	All A programs B 1 to B 4
3	B 5 to B8 C 1 to C 4

* Complexity: C; Size: 5; I/O Complexity: 4

3½	All other B C 5 to C 8 D 1 to D4
4	All other C D 5 to D 6
4½	D 7 to D 8 E 1 to E 6
5½	All other

Task 5—Assembly and Test

$$\text{Number of assemblies} = 2 + \frac{\text{Test shots}}{10}$$

Programmer time per assembly = 1 man-hour

Number of test shots equals

	Number of shots for complexity	
2 for each unit of size *plus*	3	A
	5	B
	8	C
	11	D
	15	E

Programmer time per test shot is

On console at remote center:	5 hours
On console at home location:	4 hours
Remote, using operator:	2½ hours

Task 6—Documentation

Number of additional pages to be prepared by programmer:

Title page, revision, and contents	2 (equivalent)
General description	1½
Detailed description	½ per unit of size
Operator's instructions and Halts	2 *plus* ½ per unit of size
Features and cautions	½ for each level of complexity
Other	½ for each level of complexity *plus* 4½

Therefore, additional documentation after completion of testing equals

	Pages for each level of complexity
	11 A
	12 B
1 page for each unit of size *plus*	13 C
	14 D
	15 E

Programmer time per page is .2 man-days.

	TASK 1	TASK 2										TASK 3										
COMPLEXITY	MAN-DAYS	SIZE 1	2	3	4	5	6	7	8	9	10	SIZE 1	2	3	4	5	6	7	8	9	10	C
A	$\frac{1}{2}$	$\frac{1}{2}$	1	$1\frac{1}{2}$	2	$2\frac{1}{2}$	3	$3\frac{1}{2}$	4	$4\frac{1}{2}$	5	1	2	3	4	5	6	7	8	9	10	A
B	1	$2\frac{1}{2}$	3	$3\frac{1}{2}$	4	$4\frac{1}{2}$	5	$5\frac{1}{2}$	6	$6\frac{1}{2}$	7	2	3	4	5	6	7	8	9	10	11	B
C	1	$4\frac{1}{2}$	5	$5\frac{1}{2}$	6	$6\frac{1}{2}$	7	$7\frac{1}{2}$	8	$8\frac{1}{2}$	9	3	4	5	6	7	8	9	10	11	12	C
D	$1\frac{1}{2}$	$6\frac{1}{2}$	7	$7\frac{1}{2}$	8	$8\frac{1}{2}$	9	$9\frac{1}{2}$	10	$10\frac{1}{2}$	11	4	5	6	7	8	9	10	11	12	13	D
E	$2\frac{1}{2}$	$8\frac{1}{2}$	9	$9\frac{1}{2}$	10	$10\frac{1}{2}$	11	$11\frac{1}{2}$	12	$12\frac{1}{2}$	13	5	6	7	8	9	10	11	12	13	14	E

TASK 4			TASK 5										TASK 6										
PARAMETER	MAN-DAYS	COMPL.	SIZE 1	2	3	4	5	6	7	8	9	10	SIZE 1	2	3	4	5	6	7	8	9	10	C
ALL A B1-B4	$2\frac{1}{2}$	A	2	$2\frac{1}{2}$	3	$3\frac{1}{2}$	$4\frac{1}{2}$	5	$5\frac{1}{2}$	$6\frac{1}{2}$	7	$7\frac{1}{2}$	$2\frac{1}{2}$	$2\frac{1}{2}$	3	3	$3\frac{1}{2}$	$3\frac{1}{2}$	4	4	4	$4\frac{1}{2}$	A
B5-B8 C1-C4	3	B	$2\frac{1}{2}$	3	$3\frac{1}{2}$	$4\frac{1}{2}$	5	$5\frac{1}{2}$	$6\frac{1}{2}$	7	$7\frac{1}{2}$	$8\frac{1}{2}$	$2\frac{1}{2}$	3	3	$3\frac{1}{2}$	$3\frac{1}{2}$	4	4	4	$4\frac{1}{2}$	$4\frac{1}{2}$	B
B9-UP C5-C8 D1-D4	$3\frac{1}{2}$	C	$3\frac{1}{2}$	4	$4\frac{1}{2}$	$5\frac{1}{2}$	6	$6\frac{1}{2}$	$7\frac{1}{2}$	8	$8\frac{1}{2}$	$9\frac{1}{2}$	3	3	$3\frac{1}{2}$	$3\frac{1}{2}$	4	4	4	$4\frac{1}{2}$	$4\frac{1}{2}$	5	C
C9-UP D5-D6	4	D	$4\frac{1}{2}$	5	$5\frac{1}{2}$	$6\frac{1}{2}$	7	$7\frac{1}{2}$	$8\frac{1}{2}$	9	$9\frac{1}{2}$	10	3	$3\frac{1}{2}$	$3\frac{1}{2}$	4	4	4	$4\frac{1}{2}$	$4\frac{1}{2}$	5	5	D
D7-D8 E1-E6	$4\frac{1}{2}$	E	$5\frac{1}{2}$	$6\frac{1}{2}$	7	$7\frac{1}{2}$	$8\frac{1}{2}$	9	$9\frac{1}{2}$	10	11	$11\frac{1}{2}$	$3\frac{1}{2}$	$3\frac{1}{2}$	4	4	4	$4\frac{1}{2}$	$4\frac{1}{2}$	5	5	5	E
ALL OTHER	$5\frac{1}{2}$																						

Fig. 9-1. Table of Programming Performance Standards for the IBM 1401 Tape System.

Example of Performance Standard Calculation for Average Program. (= C-5/4)

Task	*Time*
1	1 day
2	6½ days
3	7 days
4	3½ days
5	6 days (4 compiles—18 test shots—remote)
6	3½ days (18 pages added documentation)
Total	27½ days or 5½ weeks total programmer effort.

Figure 9-1 demonstrates a chart displaying the number of days required for each task. A specific chart may be developed for each installation so that the total time easily can be determined after a new program has been rated. Figure 9-1 has used the formulas developed for the IBM 1401.

Standards for the IBM 7080 Tape System

Memory Positions: 160,000 characters
Coding: Autocoder or COBOL
Average Program: C-7/8
Unit of size: One unit of size equals 10 pages of Autocoder or Autocoder equivalent, using an implicit relationship of 1 COBOL statement generating approximately 5 lines of Autocoder III coding.

Task 1—Macro-Logic

Man-days	*Complexity and Size*
2	All programs rated A
3	B 01 to B 04 C 01 to C 02
4	B 05 to B 08 C 03 to C 07 D 01 to D 03
5	All other B C 08 to C 10 D 04 to D 07 E 01 to E 02
6	All other C D 08 to D 10 E 03 to E 06
7	All other

Task 2—Micro-Logic

½ day per unit of size *plus*
1½ days for each tape unit used in excess of 4 *plus*

Man-days for level of complexity	
1	A
2	B
5	C
8	D
12	E

Task 3—Coding

Autocoder: 1½ days per unit of size *plus*

Man-days for level of complexity	
0	A
1	B
2	C
3	D
4	E

COBOL: ½ day per unit of size *plus*

Man-days for level of complexity	
0	A
1½	B
3	C
4½	D
6	E

Task 4—Desk Checking

Autocoder *Man-days*	*Complexity and Size*	COBOL *Man-days*
4	All A B 1 to B 4	3
5	B 5 to B 8 C 1 to C 4	4
6	All other B C 5 to C 8 D 1 to D 4	5
7	All other C D 5 to D 8 E 1 to E 4	6
8	All other D E 5 to E 8	7
9	All other	8

Task 5—Assembly and Test

$$\text{Number of assemblies} = 3 + \frac{\text{Test shots}}{10}$$

Time per compile = .2 man-days

Number of tests equals

1½ for each unit of size *plus*
2 for each tape unit in excess of 4 *plus*

	Number of shots for level of complexity
	4 A
	6 B
	8 C
	14 D
	18 E

Programmer time per test shot is

On console at remote center:	6 hours
On console at home location:	5 hours
Remote, using operator:	3 hours

Task 6—Documentation

	Number of pages for level of complexity
	14 A
	16 B
1½ pages for each unit of size *plus*	18 C
	20 D
	22 E

Programmer time per page = .2 man days

IBM 7080 Summary for average program (= C-7/8)

	Time in Days	
Task	Autocoder	COBOL
1	4	4
2	14½	14½
3	13½	5½
4	6	5
5	12	12 (6 compiles, 27 test shots)
6	6	6 (29 added pages)
Totals	56	47

Standards for the IBM 7090 Tape System

Memory Positions: 32,000 words
Coding: FORTRAN or FAP Symbolic
Average Program: C-7/6
Unit of size (FORTRAN or FAP Symbolic): 10 pages of coding (FAP equivalent)

Task 1—Macro-Logic. The macro-logic for the IBM 7090 is the same as that for the IBM 7080 *plus* the addition of ½ days for analysis and restatement of required formulas.

Task 2—Micro-Logic

1 day for each unit of size *plus*
1½ days for each tape unit in excess of 4 *plus*

Man-days for level of complexity	
1	A
2	B
4	C
6	D
8	E

Task 3—Coding

FAP: 2 days for each unit of size *plus*
½ day for each tape unit in excess of 4 *plus*

Man-days for level of complexity	
2	A
4	B
6	C
8	D
10	E

FORTRAN: ½ day per unit of size *plus*

Man-days for level of complexity	
1	A
2	B
3	C
4	D
5	E

Task 4—Desk Checking

FAP		FORTRAN
Man-days	*Complexity and Size*	*Man-days*
4	All A B 1 to B 4	2
5	B 5 to B 8 C 1 to C 4	3
6	All other B C 5 to C 8 D 1 to D 4	4
7	All other C D 5 to D 8 E 1 to E 4	5
8½	All other	6

Task 5—Assembly and Test

$$\text{Number of compiles} = 2 + \frac{\text{Test Shots}}{5}$$

Time per compile = ¼ man-day

Number of check-out runs equal
2 for each unit of size *plus*
2 for each tape unit in excess of 4 *plus*

Check-outs for level of complexity	
3	A
6	B
10	C
13	D
17	E

Programmer time per check-out is

On console at remote center:	5 hours
On console at home location:	4 hours
Remote, using operator:	2½ hours

Task 6—Documentation

1 page per unit of size *plus*

Number of pages for level of complexity	
8	A
9	B
10	C
11	D
12	E

Programmer time per page = .2 man days

IBM 7090 Summary for average program (= C-7/6)

	Time in Days	
Task	*FAP*	*FORTRAN*
1	4½	4½
2	14	14
3	21	6½
4	6	4
5	10½	10½ (8 compiles, 28 check-outs)
6	3½	3½ (17 pages added)
Totals	59½	43

Standards for the UNIVAC 490 Tape System

Memory Positions: 32,000 words
Coding: SPURT
Average Program: C-5/5
Unit of size: 10 pages of coding

Task 1—Macro-Logic

Man-days	*Complexity and Size*
2½	All A programs B 1 to B 4
3½	B 5 to B 8 C 1 to C 4 D 1 to D 2
4½	All other B C 5 to C 8 D 3 to D 6
5½	All other C D 7 to D 10 E 1 to E 6
7	All others

Task 2—Micro-Logic

½ day for each unit of size *plus*
1 day for each input-output unit in excess of 3 *plus*

Man-days for level of complexity	
1	A
3	B
5	C
8	D
11	E

Task 3—Coding

1 day for each unit of size *plus*
½ day for each input-output unit in excess of 3 *plus*

Man-days for level of complexity	
0	A
1	B
2½	C
4	D
6	E

Task 4—Desk Checking

Same standards as for macro-logic *plus* ½ day

Task 5—Assembly and Test

$$\text{Number of compiles} = 4 + \frac{\text{Test shots}}{10}$$

Time per compile = .2 man-days
Number of test shots equals

	Man-days for level of complexity	
	3	A
	5	B
2 per unit of size *plus*	8	C
	12	D
	16	E

Programmer time per test shot is

On console at remote center:	5 hours
On console at home location:	4 hours
Remote, using operator:	2½ hours

Task 6—Documentation

	Number of pages for level of complexity	
	11	A
	13	B
1 page per unit of size *plus*	15	C
	17	D
	19	E

Programmer time per page = .2 man-days

UNIVAC 490 Summary for average program (C-5/5)

Task	*Time in Days*	
1	4½	
2	9½	
3	8½	
4	5	
5	6½	(6 compiles, 18 test shots remote)
6	4	(20 added pages)
Total	38	

Development of Standards for Maintenance Programming

Similar standards can be developed and applied to maintenance programming except that they apply to the *change,* not to the entire program. Size is therefore measured as the size of the change, in units of 10 pages of coding (or fractions thereof). Complexity is the complexity of the change. The tasks have already been defined; standards shown here apply to the maintenance of programs written for the IBM 1401—others may be developed by the reader.

Task 1—Review of Change

Man-days for level of complexity	
½	A
½	B
1	C
2	D
3	E

Task 2—Development of Logic

		Man-days for level of complexity	
		0	A
1 day for each unit of size		½	B
or fraction thereof	*plus*	1½	C
		2	D
		3	E

Task 3—Coding and Desk Checking

		Man-days for level of complexity	
		0	A
1 day for ~~each~~ unit of size		0	B
or fraction thereof	*plus*	½	C
		1	D
		2	E

Task 4—Test and Assembly

Number of assemblies = 2
Total time = ¼ man-day
Number of test shots equals

		Test shots for level of complexity	
		1	A
1 for each unit of size		1	B
or fraction thereof	*plus*	3	C
		5	D
		7	E

Programmer time per test shot

Open shop:	5 hours
Closed shop:	3 hours

Task 5—Documentation and Turnover

		Man-days for level of complexity	
		0	A
¼ day for each unit of		0	B
size or fraction thereof	*plus*	1	C
		2	D
		2	E

A second approach to maintenance programming standards is to regard a change as a fractional part of the entire program. The entire program is rated and the standard calculated as if it were a development program. The change effort is then estimated as a percentage of the overall program and this percentage applied to produce the standard. For example:

A 1401 program rated as a D-6 has to be modified to incorporate an added field in the detail tape record and to show the field on the output report. The change is approximately 10% of the entire program.

If the entire program were to be rewritten, the following standard would apply (from Figure 9-1, page 257):

Task 1	1½
Task 2	9
Task 3	9
Task 4	4
Task 5	7½
Task 6	4
Total	35 days

10% of the change represents an effort of 3½ man-days.

Using the detailed standards for maintenance programming the following would apply, if the change is rated as B-1.

Task 1	½
Task 2	1½
Task 3	1
Task 4	¾
Task 5	¼
Total	4 man-days

Note that the two methods give slightly different answers. This is due to the difference in approach; rewriting 10% of a program really takes more than 10% of the total time.

DEVELOPMENT OF A PROGRAMMING SCHEDULE

The first step in scheduling is to rate the programs to be developed and maintained on the basis of complexity, size and input-output complexity.

Rating Complexity.—The most experienced programmer or supervisor should rate the program based on the system flowchart. The same person should do all of the rating so that all programs are rated in the same manner.

Rating Size.—If possible, the same person who rates the complexity

should estimate the number of pages of coding. This rating can easily be checked against the number of pages of coding actually produced. If there is consistent error in the program size, all future programs should be corrected for this error or the estimating method reviewed.

Rating Input-Output Complexity.—This rating, preferably accomplished by the same person, is a mechanical count of the number of input and output units or tapes, which the program uses. Alternate tape files, which are called into action through a flip-flop or servo swap routine should not be counted. The objective is to measure the number of distinct *files* which the program must control.

After the rating has been completed, the man days required for each of the tasks can be calculated. A table may be used as an aid, such as illustrated in Figure 9-1, page 257, or a computer program may be developed to calculate the values automatically. In either case, the calculations should keep the values for each task completely separate, so that evaluation can be made by program, programmer and function.

After the values have been calculated, it is a simple matter to establish a development schedule. This can be a simple bar chart, assigning the work to specific programmers, or a complex computer program, using the "PERT" technique of critical path scheduling.

Figure 9-2 shows the output of a computer program which has calculated the time and cost of writing a series of programs for the Univac 1107. It has also calculated the machine time necessary for testing, key-punching cost, and the cost of operating the system for testing and compiling. The costs are broken down by system code. Man days have been converted into man-years in the subtotals simply by dividing by 241.

Figure 9-3 is a typical form used to show an individual programmer's schedule. All programmers' schedules are maintained by the programming supervisor, to enable review of progress each period.

Figure 9-4 is a slightly different type of schedule, set up by program and used as a historical record of progress. The complexity rating of each program is indicated at the top, and the number and schedule of expected days is indicated in the body of the form. This form also records information on the number of tests and the amount of machine time used for testing the program. This information will be compared to the standards and used in future estimating.

GATHERING PERFORMANCE DATA

With an accuracy equal to the data obtained for equipment performance (Chapters VI and VIII), data must be gathered about the actual performance of the programming. Unfortunately, this involves the same

THE DIEBOLD GROUP, INC. -LOCKHEED GEORGIA STUDY -FINANCIAL PROGRAMS - DETAIL LIST 0 8/24/62 PAGE 69

PROG# -JOB- RPT-F3	SYS	RATE	REDO MAN DAYS	--PROGRAMMER CONVERSION MAN-DAYS-- TASK 1	TASK 2	TASK 3	TASK 4	TASK 5	TASK 6	TOTAL PROGR DAYS	EFFECTV PROGR DAYS	1107 OPER DAYS	KEY PUNCH DAYS	# OF SHOTS	MACH HRS	DIRECT LABOR COSTS	MACHINE COSTS	TOTAL COSTS
4707	047	F 31/11	31.9	6.0	44.5	28.9	25.1	13.7		118.2	118.2	8.3	1.9	67	19.3	$5,136	$8,028	$13,164
4708	047	B 06/05	8.3	3.0	10.5	5.7	5.5	4.5		29.2	29.2	2.0	.3	16	2.0	$1,281	$832	$2,113
4709S	047	/01				1.0				1.0	1.0	.2		1	.1	$70	$41	$111
4710	047	C 22/07	21.1	4.0	29.5	19.2	16.3	9.5		78.5	78.5	5.6	1.3	45	10.5	$3,421	$4,368	$7,789
TOTAL	MAN YRS		.25	.05	.35	.23	.19	.11	.00	.94	.94	.06	.01	130	32.0	$***9,978	$*13,310	$**23,288
7775	048	C 08/05	11.4	4.0	14.5	8.2	7.5	6.0		40.2	40.2	2.6	.5	21	2.8	$1,752	$1,164	$2,916
7776S	048	/01				1.0				1.0	1.0	.2		1	.1	$70	$41	$111
7777	048	B 06/03	8.0	3.0	10.0	5.5	5.5	4.5		28.5	28.5	2.0	.3	16	2.0	$1,252	$832	$2,084
7787S	048	/01				1.0				1.0	1.0	.2		1	.1	$70	$41	$111
7788	048	C 08/06	11.7	4.0	15.0	8.5	7.5	6.0		41.0	41.0	2.6	.5	21	2.8	$1,785	$1,164	$2,949
7792S	048	/01				1.0				1.0	1.0	.2		1	.1	$70	$41	$111
7795	048	09/03	15.3	6.0	19.0	10.7	11.5	8.2		55.4	55.4	3.8	.5	31	4.0	$2,408	$1,664	$4,070
8796S	048	/01				1.0				1.0	1.0	.2		1	.1	$70	$41	$111
8797	048	C 04/04	8.0	3.0	10.0	5.0	5.1	5.0		28.1	28.1	1.8	.2	15	1.6	$1,228	$665	$1,893
8798S	048	/01				1.0				1.0	1.0	.2		1	.1	$70	$41	$111
8799	048	C 08/04	10.6	3.0	14.0	8.0	7.5	6.0		38.5	38.5	2.6	.5	21	2.7	$1,682	$1,123	$2,805
TOTAL	MAN YRS		.26	.09	.34	.21	.18	.14	.00	.98	.98	.06	.01	130	16.4	$**10,455	$**6,817	$**17,272
4712S	049	/01				1.0				1.0	1.0	.2		1	.1	$70	$41	$111
4715	049	D 15/09	19.0	5.0	25.5	15.4	13.3	8.7		67.9	67.9	4.5	.9	36	7.2	$2,948	$2,995	$5,943
4734S	049	/01				1.0				1.0	1.0	.2		1	.1	$70	$41	$111
4735	049	B 11/02	11.2	3.0	15.0	9.2	8.5	5.7		41.4	41.4	3.0	.6	24	3.6	$1,815	$1,497	$3,312
TOTAL	MAN YRS		.12	.03	.16	.11	.09	.05	.00	.46	.46	.03	.00	62	11.0	$***4,908	$**4,574	$***9,477
3752S	050	/01				1.0				1.0	1.0	.2		1	.1	$70	$41	$111
3753	050	D 12/08	16.8	5.0	22.0	13.0	11.7	8.0		59.7	59.7	4.0	.7	32	6.1	$2,593	$2,537	$5,130
3754S	050	/01				1.0				1.0	1.0	.2		1	.1	$70	$41	$111
3755	050	B 05/04	7.3	3.0	9.0	4.7	4.7	4.2		25.6	25.6	1.7	.3	14	1.6	$1,126	$665	$1,791
TOTAL	MAN YRS		.10	.03	.12	.08	.06	.05	.00	.36	.36	.02	.00	48	7.9	$***3,859	$**3,284	$***7,143
1746S	051	/01				1.0				1.0	1.0	.2		1	.1	$70	$41	$111
1747	051	C 08/05	11.4	4.0	14.5	8.2	7.5	6.0		40.2	40.2	2.6	.5	21	2.8	$1,752	$1,164	$2,916
1748S	051	/01				1.0				1.0	1.0	.2		1	.1	$70	$41	$111
1749	051	C 11/07	14.0	4.0	18.5	10.9	9.3	6.7		49.4	49.4	3.2	.6	26	5.0	$2,148	$2,080	$4,228
TOTAL	MAN YRS		.10	.03	.13	.08	.06	.05	.00	.38	.38	.02	.00	49	8.0	$***4,040	$**3,326	$***7,366
4711	052	B 04/02	6.7	3.0	8.0	4.0	4.3	4.0		23.3	23.3	1.6	.2	13	1.4	$1,026	$582	$1,608

Fig. 9-2. Computer Analysis of Performance. (*Courtesy*, Lockheed-Georgia Company, A Division of Lockheed Aircraft Corporation)

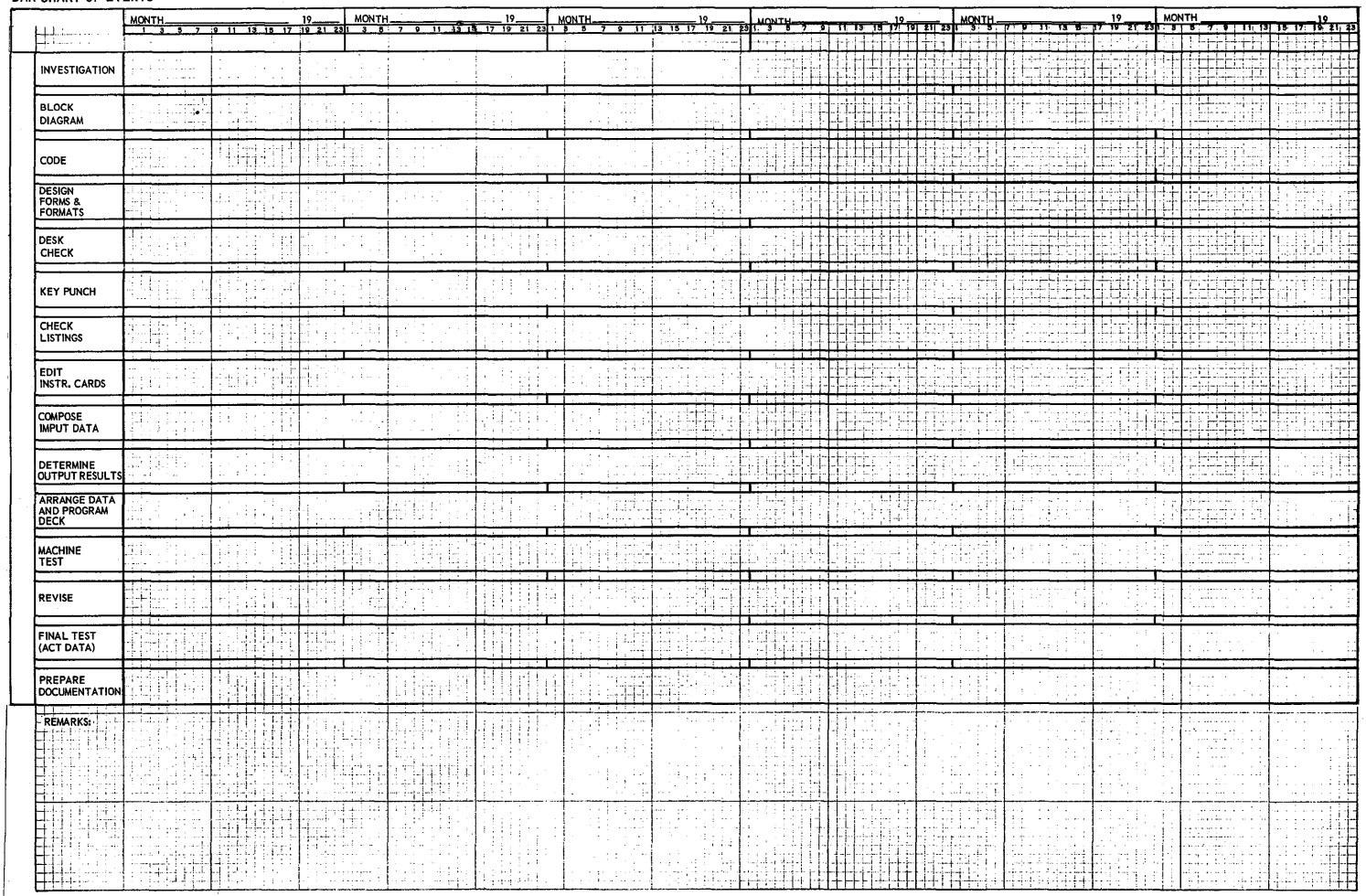


Fig. 9-3. Bar Chart of Events. (*Courtesy*, Atlantic Refining Company)

Bowery Savings Bank - Electronic Data Processing Division

PROGRAM PLANNING AND PROGRESS CHART

PROGRAM NAME _____

PROGRAM NUMBER _____ APPLICATION _____

PROGRAMMING EFFORT ESTIMATE _____ DAYS RUN TIME ESTIMATE _____ HOURS

DEGREE OF PROGRAMMING DIFFICULTY INVOLVED IN THIS PROGRAM _____

SECTION I TO BE COMPLETED PRIOR TO THE FIRST TEST SESSION

	BY	ESTIMATED			ACTUAL			COMMENT
		START DATE	COMPL. DATE	ELAPSED DAYS	START DATE	COMPL. DATE	ELAPSED DAYS	
GENERAL STATEMENT OF PROBLEM								
LOGIC DIAGRAM								
I/O DESIGN INCLUDING FORMS								
DETAIL BLOCK DIAGRAM								
CODING								
PREPARATION OF TEST DATA								
PRELIMINARY DOCUMENTATION								
KEYPUNCH, EAM AND DESK CHECK								
TOTALS								VARIANCE IN DAYS (EST.)-(ACT.)

SECTION II TO BE ACCOMPLISHED DURING AND AFTER TESTING

	BY	ESTIMATED			ACTUAL			COMMENT
		START DATE	COMPL. DATE	ELAPSED DAYS	START DATE	COMPL. DATE	ELAPSED DAYS	
FIRST ASSEMBLY								
DESK CHECK								
ASSEMBLY TEST AND DEBUG								
FINAL DOCUMENTATION								
CONVERSION PROCEDURES (IF ANY)								
TOTALS								VARIANCE IN DAYS (EST.)-(ACT.)

PROGRAM TESTING LOG

DATE																								
MACHINE HOURS																								
MAN HOURS																								

REMARKS: _____

ER 2590X

Fig. 9-4. Program Planning and Progress Chart. (*Courtesy*, The Bowery Savings Bank)

detailed record-keeping, although it is now applied to the programming staff, and cannot be obtained mechanically. This is a difficult problem. Programmers often do not want, and will not take the time, to keep

minute records of their own activities. An argument frequently heard against performance standards is "How can I ask my programmers to keep any more records than they now have to keep, when I have difficulty getting them to endorse their paychecks?" Yet the entire concept of management control demands enforcement of these simple requirements.

Performance measurement data must be obtained by program, by task, and by programmer. Three kinds of performance data must be obtained: time, quality, and validity of standards.

Measurement of Quality.—The measures which are established to evaluate the quality of performance are partially subjective, and partially factual. Subjective measures include a rating of the documentation, and an evaluation of the logical completeness. These are discussed in more detail in a later section.

Measuring the Validity of Standards.—Data gathering should include, for comparison to the standards:

- Number of test shots (also a measure of quality)
- Number of compilations
- Number of pages of documentation
- Program size

Measuring Programming Time.—The time spent by each programmer on each task can be obtained in a number of ways:

1. From a report of progress by program, requiring the programmer to record weekly the time spent on each task. If the programmer has worked on 10 programs, he must submit 10 such reports; the totals of all, plus any "indirect" time, must add to total paid time. Figure 9-5 shows such a report. The programmer fills out the days or fractions thereof spent in any one week on any task of the program, and also estimates each week the number of days which he believes are required to complete task. The programming manager accumulates the days spent to date, and compares these with the days scheduled for each task. The programmer's estimate is used only to signal the need for review of status when the sum of the days needed to complete and the days spent exceeds the total days scheduled. Figure 9-6 is a variation of this form, set up for key punching.

2. From a weekly report on which the programmer records all of his time. Distribution by program must then be done separately. Figures 9-7, 9-8, and 9-9 demonstrate this type of reporting, one closer to normal production reporting methods.

3. From a "Program Follower Ticket": When a program is assigned, a basic record and recording form is created to stay with the program until completion. This method enables detailed evaluation by program,

BOWERY SAVINGS BANK–E.D.P. DIVISION

WEEKLY RECORD OF PROGRESS

PROGRAMMER ..... WEEK ENDING .....

PROGRAM NAME .....

PROGRAM NO. ..... APPLICATION .....

FUNCTION	DAYS SPENT THIS WEEK	DAYS SPENT TO DATE	DAYS REQ. TO COMPLETE	SCHEDULED MAN-DAYS
1. Problem Definition				
2. Macro-Logic				
3. I/O Layouts				
4. Micro-Logic				
5. Coding				
6. Desk Checking				
7. Test Data				
8. Test & Revise				
9. Documentation				
TOTALS				

REMARKS: _____

ER 2592x

Fig. 9-5. Weekly Progress Report by Program, by Programmer, and by Task. (*Courtesy*, The Bowery Savings Bank)

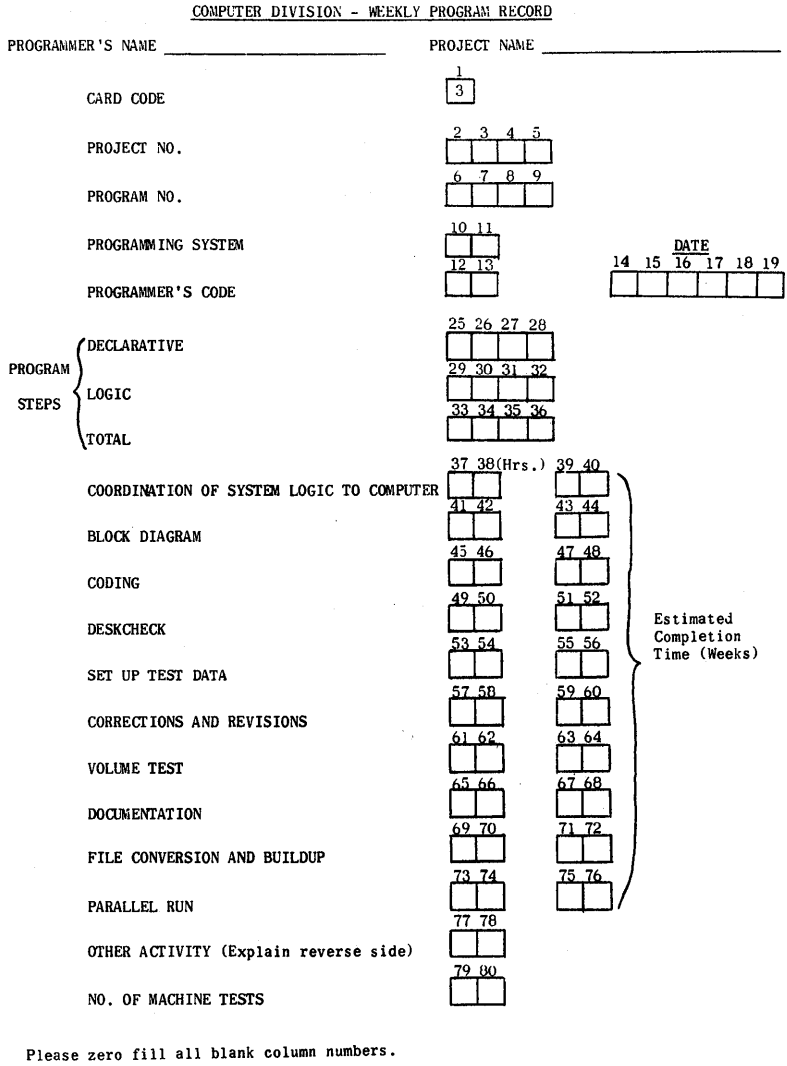


Fig. 9-6. Weekly Program Record. (*Courtesy*, Atlantic Refining Company)

but does not permit easy reconciliation with the total hours worked by any employee. To stay current it must remain with the work-in-progress; the manager, to make a status evaluation, must therefore go to each work station and assume that the latest figures have been recorded. This kind of report is shown as Figure 9-10*a*, and a variation is shown as Figure 9-10*b*.

There are, of course, other methods of reporting and recording time

DATA PROCESSING PLANNER'S WEEKLY REPORT

Emp. No.	Grp.	Name	Week Ending	Tot. Hrs.	S.T. Hrs.	O.T. Hrs.

Date	Assign-ment	Code B	Code S	Hrs.	O T	Remarks	Date	Assign-ment	Code B	Code S	Hrs.	O T	Remarks

01 Problem Definition
02 Phase III ½
03 Systems Flow Diagram
04 Program Flow Diagram
05 Detail Logic Diagram
06 Customer Contact
07 Coding
08 Desk Checking
09 Test Data Preparation
10 Test Checking
11 Job Instructions
12 EAM Planning
13 Console Assistance
14 Assisting Operations
15 Assisting Dispatch
16 Instruction
17 Non-Job Related
18 Supervision
19 Tardy—Short Time
20 Absent

Suffix Codes

1. New Job—Recurring
2. Special Request
3. Reprogramming
4. Revision—Customer
5. Revision—Departmental
6. Program Error Corr.
7. Operations Error Corr.

Fig. 9-7. Weekly Reporting by Programmer. (*Courtesy*, Lockheed-Georgia Company, A Division of Lockheed Aircraft Corporation)

spent by program, task, and programmer. The unit of measurement used, however, is sufficiently inexact so that a more formal means is not required generally. The smallest unit usually used is one-half man-day, occasionally ¼ man day. It is neither necessary nor meaningful to collect performance data in smaller units, unless this information is also used as a basis for payroll. In this case a time clock or other measurement device may be used.

EMPLOYEE WEEKLY LABOR DISTRIBUTION SHEET OF

EMPLOYEE NO.	NAME OF EMPLOYEE	HOME DEPT. & FUNC.	SHIFT	WEEK ENDING
☐ HOURLY ☐ FLAT SALARY	AUTHORIZED SHIFT START END	WORK DEPT.	SIGNATURE OF EMPLOYEE	

TO BE FILLED-IN BY EMPLOYEE AND KEPT ON HIS DESK AVAILABLE AT ALL TIMES TO TIMEKEEPER

ACCOUNT OR WORK ORDER	STA C.C.	WAP / PEJO - FRR / IR - TOSN (ARTICLE, ITEM)	EID	BENEFIT DEPT / RESP DEPT	MON. START/STOP ★	TUES. START/STOP ★	WED. START/STOP ★	THURS. START/STOP ★	FRI. START/STOP ★	SAT. START/STOP ★	SUN. START/STOP ★	TOTAL HOURS
★ TOTAL ELAPSED TIME PER JOB TO BE COMPUTED BY TIMEKEEPER			TOTAL HOURS									
POSTED AND INITIALED BY TIMEKEEPER												

CONVAIR | ASTRONAUTICS FORM A198 (REV. 12-58)

Fig. 9-8. Weekly Labor Distribution. (*Courtesy*, General Dynamics/Astronautics Division)

WORK-IN-PROGRESS REPORT

GROUP NUMBER (2 3)	NAME (4 5 6 7)	FUNCTION (8)	WK. NO. (9 10)	WEEK ENDING	MONTH (11 12)	DAY (13 14)	YR (15)

AWO / IND CHG OR EWO (16 18 / 19 22)	WAP (23 26)	JOB NO. (27 31)	BENEFIT GROUP (32 35)	P L (36)	DATE: BEGIN (37 41)	DATE: SCH. COMP. (42 46)	DATE: EST. COMP. (47 51)	MAN HOURS: EST'D (52 55)	MAN HOURS: THIS WEEK (56 58)	MAN HOURS: REQ'D TO COMPLETE (59 62)	EST'D 7000 HOURS (63 65)	C W (66)	PROGRESS GRAPH Δ B, C, D, E, L, P, S, T, V (67 69 70 72 73 75 76 78)	* (79 80)

CHECKED BY:

* ON TITLE LINE ENTER JOB NUMBER (27-31) AND JOB TITLE (32-78)
11 = JOB DELETE
12 = DELETE AND ADD TO HISTORY FILE
13 = ADD JOB ENTRY LINE
23 = ADD TITLE LINE
14 = CHANGE ENTRIES
24 = CHANGE TITLE

A4038 (2-62)

Δ B = BEGIN C = COMPLETE D = DEVELOP E = ESTIMATED = (BLANK) L = LEAVE T = TRAINING P = PRODUCTION S = SCHEDULED V = VACATION

Fig. 9-9. Work-In-Progress Report. (*Courtesy*, General Dynamics/Astronautics Division)

PROGRAM No._____ APPLICATION _____

PROGRAM NAME _____ ASSIGNMENT DATE _____

TASK NO	TASK	SCHEDULE	(PERIOD 1)						
1	MACRO-LOGIC		(PAYS)						
	ASSIGNED TO:	DATE:	(ANALYST Date)						
2	MICRO-LOGIC								
	ASSIGNED TO:	DATE:							
3	CODING								
	ASSIGNED TO:	DATE:							
4	DESK CHECKING								
	ASSIGNED TO:	DATE:							
5	TEST AND ASSEMBLY								
	ASSIGNED TO:	DATE:							
6	DOCUMENTATION								
	ASSIGNED TO:	DATE:							
5a	NUMBER OF TEST SHOTS — MINUTES								
	DATES:								

Fig. 9-10*a*. Program Status.

EVALUATION AND USE OF PERFORMANCE DATA

The astute manager is now in a position to analyze the collated performance information, and use it for positive management control. The objectives to be met in analysis and evaluation may vary, but generally include the following:

1. Progress Reporting.—By measuring overall "efficiency" from overall performance, it is simple to determine the exact status of the entire development program, as well as the completion date of particular systems, applications, and individual runs. This overall efficiency factor can be used to modify the overall schedule and all future planning.

2. Budgetary Control.—If the standard proves effective the time and cost required to develop the remainder of the program can be firmly established.

3. Personnel Evaluation.—The most common reason for performance evaluation is to enable an unbiased evaluation of the members of the

PROGRAM STATUS

APPLICATION _____ PROGRAM NUMBER _____ PROGRAM NAME _____ WEEK ENDING _____

PROGRAMMER / PROGRAMMER	ORIGINAL SCHEDULE START	ORIGINAL SCHEDULE COMPLETE	ORIGINAL ESTIMATED MAN-DAYS REQUIRED	CUMULATIVE MAN-DAYS EXPENDED THRU LAST WK.	MAN-DAYS EXPENDED THIS WK.	ESTIMATED % COMPLETED NOW	REVISED SCHEDULE START	REVISED SCHEDULE COMPLETE	REVISED ESTIMATED MAN-DAYS REQUIRED
ANALYSIS OF PRESENT OPERATION & REPORT DATA									
PROGRAM DEFINITION									
BLOCK DIAGRAM									
CODE									
DESK CHECK									
PREPARE TEST DATA									
PREPARE RUN BOOK									
TEST									

PROGRAMMER TO PREPARE REVISED SCHEDULE WHEN ORIGINAL SCHEDULE CANNOT BE MAINTAINED.

NUMBER INSTRUCTIONS WRITTEN: CUMULATIVE _____ THIS WK. _____ REMARKS:

PROGRAM ASSEMBLY:

	MEMORY LOCATION FROM	MEMORY LOCATION TO	MEMORY REQUIREMENTS	NUMBER INSTRUCTIONS
1. OA				
2. INSTR				
3. DC				
4. SUBROUTINE				
5. IOCS				
6. LITERALS				
7. PATCHES				
TOTAL				

PREPARED BY PROGRAMMER _____

CHECKED BY SR. PROGRAMMER _____

APPROVED BY SUPERVISOR _____

AB112 (3-62)

Fig. 9-10*b*. Program Status. (*Courtesy*, General Dynamics/Astronautics Division)

staff. This is far preferred to intuitive evaluation, which tends to favor the extroverted programmer.

4. Functional Specialization of Personnel.—One of the most interesting byproducts of the use of task-oriented standards is the ability to recognize functional specialization. A number of programmers prefer program testing, but almost as many consider machine operation, memory print evaluation, and all other tasks associated with testing demeaning, and prefer to concentrate on logical analysis. Others prefer coding, and some the rapid production of good documentation.

The development of task-oriented performance standards tends to show which programmers are most capable in each task. As a result, management may decide to establish "functional" teams, consisting of a programmer skilled in logical analysis, a good coder, a good tester, and a junior member responsible for documentation. This may prove quite economical even though communications problems are increased.

5. Program Assignment.—The use of performance standards allows accurate estimation of the time needed to complete a task. If a program is required before the standard date, it is wise to assign a programmer whose efficiency is greater than standard. Similarly, an evaluation of the total time necessary to complete a series of programs may lead to the important, but often undetected, choice of the programs to be started

first. This is important in a development program where the total load of required programs exceeds the time available before machine installation. Rather than eliminate the documentation function, at great risk and cost, it may be possible to delay the development of programs not immediately required, such as those to be run annually.

6. Setting Meaningful Delivery Dates.—The use of effective performance standards can assist in pinpointing a realistic equipment delivery date long before the system is shipped, because the date of completion of programming will be known.

Evaluation Methods

To satisfy the objectives outlined above, and to provide sufficient information for management action, the following analytical factors should be derived from performance measurement data:

- Overall departmental efficiency
- Overall departmental efficiency by task
- Programmer efficiency
- Programmer efficiency by task

The following procedure may be used to record and retain the analytical data:

1. Establish a program fact sheet, illustrated in Figure 9-11. The fact sheet summarizes all program data by task, and assists in evaluation of the performance of the programmer(s) assigned to the task.
2. Summarize performance by programmer for each task performed. This is illustrated as Figure 9-12.
3. Summarize departmental performance by task, illustrated in Figure 9-13. Part I of that form shows work completed, with efficiency rated. Part II shows the work to be completed, applies current efficiency to it, producing a tentative completion date.
4. From the summaries, obtained the factors necessary to determine the efficiencies, and the progress to date, as shown on the illustrations.

These summaries can be made up as frequently as desired. The approach depends on the urgency of installation; if the equipment is scheduled to arrive in the next few months, a frequency of two weeks is not unreasonable. If the installation is in operation, with limited new development, monthly evaluation will probably suffice. The summary by programmer can be made more frequently, if desired, especially in the early stages of the installation, when the effectiveness of the staff is still below the operating standard. In this case, the effectiveness by task can be plotted on a typical "learning" curve, such as has been

PROGRAM NAME Purchase order			PROGRAM No. T04D			
APPLICATION Inventory	DATE ASSIGNED: 3/5/63		RATING	D	7	4

PROGRAMMER	PORTION / TASK	PROGRAMMER	PORTION / TASK
A. J. Smith – Sr P.	Task 1 + 2	C. —	
B. T. Jones	all others	D. —	

TIME PERFORMANCE

TASK	BY-LETTER	SCHEDULED	ACTUAL	DIFFERENCE	% EFFICIENCY
TASK 1 – MACRO-LOGIC	A	1½	1	+ ½	150
TASK 2 – MICRO-LOGIC	A	9½	9	+ ½	106
TASK 3 – CODING	B	10	9	+ 1	111
TASK 4 – DESK-CHECKING	B	4½	5	– ½	90
TASK 5 – TEST AND ASSEMBLY	B	8½	9	– ½	95
TASK 6 – DOCUMENTATION	B	4½	3½	+ 1	128
TOTALS:		38½	36½	+ 2	106
BY PROGRAMMER: A		11	10	+ 1	110
B		27½	26½	+ 1	103
C					
D					

VALIDATION

FACTOR	SCHEDULED	ACTUAL	DIFFERENCE	REASONS
SIZE	7	77	—	✓
PAGES OF DOCUMENTATION	21	17	4	Less Prog Halts
TEST SHOTS	25	28	3	89% E-1
COMPILATIONS	5	4	1	120% E-2
OTHER:				

QUALITY RATINGS: Logic OK – Test 89/120

ERRORS AFTER PRODUCTION: 1 – 6/28/63

DOCUMENTATION QUALITY – 85/100 OK

SUPERVISOR APPROVAL: Doe DATE 6/1/3

Fig. 9-11. Example of Program Fact Sheet.

illustrated in Figure 9-14. This reflects staff efficiency by task, and may be made up for each programmer or for the entire department. It can be maintained for as long as is necessary in order to pinpoint reductions in efficiency caused by changes in operation, methods standards, morale, and the like. The graph clearly indicates that the particular programmer is much more proficient at coding, desk checking and documentation than at testing; this has held true throughout the learning process, but may change. Certain testing techniques require a great deal more experience than the other tasks involved.

PROGRAMMER: T. Jones SUPERVISOR: Doe ACCOUNTING PERIOD: 12/63

WORK COMPLETED IN PERIOD				MANDAYS SCHEDULED								
PROGRAM	RATING	INITIAL STATUS & %	FINAL STATUS & %	T1	T2	T3	T4	T5	T6	TOTAL	TEST SHOTS	COM-PILES
T12M	B-3/2	T2-100%	T4-100%			4	2½			6½	—	—
T04D	D-7/4	T2-100%	T3-50%			5				5	—	—
T03M	C-4/4	T4-100%	T5-80%					4½		4½	14	4
T04M	A-3/2	T3-50%	T5-100%			1½	2½	3			9	3
TOTALS						10½	5	7½		23	23	7
ACTUAL TIME SPENT * BY TASK						9	4½	8½		22	25	6
EFFICIENCY						117	111	88	—	104	92	116
LAST PERIOD					87	112	—	—	108	106	—	—

Fig. 9-12. Example of Programmer Performance Summary.

PART I - WORK COMPLETED ACCOUNTING PERIOD: 12-63

PROGRAM	RATING	INITIAL STATUS & %	FINAL STATUS & %	PROG.	T1	T2	T3	T4	T5	T6	TOTAL	TEST SHOTS	COM-PILES
T01D	B 3/2	0	0								0		
T02D	A 4/2	0	0								0		
T03D	C 5/2	T6-100%									0		
T04D	D 7/4	T2-100%	T3-50%	Jones			5				5		
T05D	B 3/3	T0	T3-100%	Smith	1	3	3				7		
DAILY TOTALS		—	—	—	1	3	8				12		
T12A	E 7/3	T3-100%	T4-100%	Smith				5½			5½		
T13A	B 3/5	T1-100%	T6-100%	Doe		3½	4	2½	3½	3	16½	11	3
ANNUAL TOTALS					5	16½	11	8	6½	3	50	21	6
GRAND TOTAL					29	47⅓	43	26	39	12	196⅓	137	36
ACTUAL DAYS SPENT					36	52	39⅔	21	41	11⅔	201	152	34
EFFICIENCY					81	92	108	122	95	104	94	90	106
LAST PERIOD					84	91	105	120	99	103	93	92	105

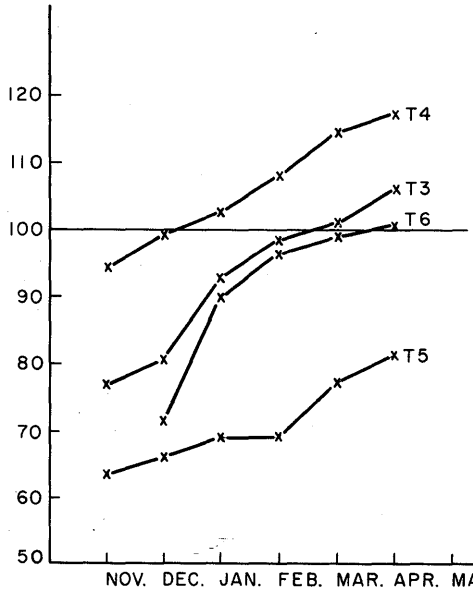
Fig. 9-13*a*. Example of Installation Performance Summary: Work Completed.

PART II: WORK TO BE COMPLETED — ACCOUNTING DATE: March 1, 1963

PROGRAM	RATING	INITIAL STATUS & %	FINAL STATUS & %	PROG.	T1	T2	T3	T4	T5	T6	TOTAL	TEST SHOTS	COMPILES
T01D	B 3/2	0	0	N	1	3 1/2	4	2 1/2	3 1/2	3	17 1/2	11	3
T02D	A 4/2	0	0	N	1/2	2	4	2 1/2	3 1/2	3	15 1/2	11	3
T04D	D 7/4		T3 - 50%	Jones			5	4 1/2	8 1/2	4 1/2	22 1/2	23	5
T05D	B 2/3		T3 - 100%	Smith				3	3 1/2	3	9 1/2	9	3
DAILY TOTALS					1 1/2	5 1/2	13	12 1/2	19	13 1/2	65	54	14
T12A	E 7/3		54 - 100%	Smith					9 1/2	4 1/2	14	29	5
ANNUAL TOTALS							17 1/2	11	36 1/2	29	94	112	23
GRAND TOTALS					17	38 1/2	83	76 1/2	178	112	505	482	48
(FROM PART I) CURRENT EFFICIENCY					81	92	108	122	95	104			
NET DAYS TO GO					21	42	77	63	187	108	498		
ANNUAL PROGRAM NET DAYS TO GO							16	9	38 1/2	28	9 1/2		
QUARTERLY PROGRAMS NET DAYS TO GO						12	13 1/2	16	29	23 1/2	94		

TOTAL NET DAYS: 498
NO. OF PERSONNEL: 9
LAPSED DAYS: 55 1/2
CURRENT COMPLETION EST: May 14, 1963
COMPLETION IF ELIMINATED: ANNUAL May 1, 1963 | QUARTERLY April 15, 1963

Fig. 9-13*b*. Example of Installation Performance Summary: Work To Be Completed.



PROGRAMMER: T. Jones

PERIOD	T1	T2	T3	T4	T5	T6
Nov 62			76	94	63	
Dec 62			80	99	65	71
Jan 63			93	103	68	90
Feb 63			98	107	68	96
Mar 63			101	115	76	98
Apr 63			106	118	81	101

Fig. 9-14. Example of Learning Curve for Programmer.

MANAGEMENT ACTION

Performance evaluation will provide management with basic information about the operation. Management might learn, for example, that

- The installation will not be completed in time
- A particular programmer is consistently under or over standard
- A particular programmer is consistently over standard in some functions and under standard in others
- The cost of a given change has been excessive
- The cost of a certain requirement of a new system could be markedly reduced by the elimination of some simple items
- The staff is functioning poorly under new supervision, salary, or other conditions, as shown in the performance graph.

The management action that results can be measured with the same system. The kinds of action which improve operations include the following:

1. *Increase or Decrease Staff.*—Performance measurement may show the need for a more realistic schedule. Its completion dates can be modified by varying manpower.

2. *Provide Incentives.*—If the entire staff is operating below or close to standard, it may be possible to improve morale and productivity by providing direct incentives. With a direct measure of performance, it is possible to demonstrate that an increase in productivity will lead to direct rewards.

3. *Develop Functional Specialization.*—By knowing the tasks or areas in which each member of the staff excels, it is possible to set up a meaningful functional approach: establishing teams of people with complementary capabilities.

4. *Reduce the Immediate Workload.*—This can be done by eliminating programs with low frequencies, such as annual or quarterly runs, or by eliminating a section of the application that can continue to be done in the existing manner.

5. *Speed Up Training.*—If the efficiency of newer employees is too low an increase may be forced by intensifying training either in the normal training program or through extra on-the-job training.

6. *Alter Procedures.*—Low productivity may point to inefficient procedures or to negative external influences. One solution may be to alter the procedures used. Another solution may be to change the work place, or the environment.

7. *Lower Quality Standards.*—Although this is not recommended, it

may be found necessary to reduce quality requirements. This may be done by reducing the documentation specifications, or by reducing the requirements of each task. Even though this is the most frequent action taken, it is never the most economical.

8. Delay Equipment Delivery.—Although the staff usually prefers to get the machine in the house as soon as possible, providing more test time, if the schedule shows that the programs will not be ready it *never* pays to bring the equipment in early. It is almost impossible to accomplish conversion while incomplete programs are being tested. The machine should not be brought in until a sufficient number of applications are completely tested and documented, to provide an economic breakeven point.

Obviously, many other management actions can be taken. They all depend to a very large extent on the availability of accurate and timely information; this is one of the major needs of data processing management in both the planning and operating stages.

ESTABLISHING CONCURRENT MEASURES OF QUALITY

The performance measurements so far outlined cover only the satisfactory completion of programmers' tasks in a given period. These standards place a premium on quantity: the ability to produce much material in a short time will enable a programmer to reach or exceed standard. They do require that the program produced must be accurate, i.e., that is it must create the specified output from the indicated input, with all necessary controls.

There has been no emphasis on quality up to this point. It is possible to produce an inefficient program while exceeding standard; a carefully thought out program might be more efficient, but would not allow the programmer to exceed standard. Similarly, the writing of sloppy, incomplete documentation would provide an opportunity to exceed standard on this task without meriting the indicated efficiency rating.

It is necessary to establish equally rigorous measures of quality, concurrent with measures of quantity. Quality measures should always indicate the minimum standard to be attained for a given quantity. A number of points of qualitative review should be established, among which are the following:

Program Efficiency

The measure of program efficiency is a function either of the execution time of the program, or of the total memory space it uses. Most often it is a direct function of execution time, which in itself may depend

on the memory space used. Techniques used to evaluate program quality include

- Comparison of actual running time against the original estimate
- Comparison of actual size against the original estimate
- Review of the program for areas capable of optimization

Running Time.—The systems analyst originally estimates program running time as the sums of input time, output time and process time for "active" and "inactive" records, (Figure 3-14*a*, page 62, shows a sample timing calculation) taking into account possible overlap, multiple processing, and time sharing. This estimate, at a given level of volumes, can be used to measure program efficiency, by running the program at the same level (which should be a minimum of twenty minutes to include all possible conditions and provide accurate timing), and the efficiency is calculated as:

$$\frac{\text{Estimated running time}}{\text{Actual running time}}$$

On the basis of the figure obtained, the quantity standard previously obtained can be adjusted in three ways. The first is to divide the quality rating by this figure. The second is to multiply the two ratings together, thus reducing the higher standard by the lower one. (A programmer who has produced a program at 130% of standard with an efficiency of 90% would have an overall rating of 117%; a programmer preparing a program at 100% of standard with an efficiency of 120% would have an overall rating of 120%.) The third method is to set a minimum quality standard which must be achieved before the quantity rating is considered acceptable, thus, if the minimum quality standard is 95%, all programs which fall below this requirement will have to be rewritten or modified to bring "efficiency" up to the minimum. After the rewriting, if the quality standard has been reached, the total of writing and rewriting time will be considered in determining the quantity rating.

Size.—In a small-core machine, the size of the program may be a reasonable measure of its efficiency. An efficiency factor is derived by relating the estimated size to the final size:

$$\frac{\text{Estimated size}}{\text{Final size}}$$

The factor developed can be used as above to adjust the quantity standard, but because the measure of size is less accurate, it should be given

a lower weight; a weighting of 4:1 in favor of the performance standard is suggested.

Program Review.—The third method for measuring qualitative efficiency is to have the finished program reviewed by a senior programmer. If it is assumed that the program has been 100% optimized and the reviewing programmer is given a fixed amount of time for each review, a factor can be established for each instruction and memory word or character that the senior programmer can eliminate. For example, if a rule is established that the same programmer will always perform the quality review, spending one hour for each unit of program size, the number of program words he eliminates can be multiplied by a factor and the total subtracted from the quantity rating.

Documentation Quality Evaluation

Output documentation should always be reviewed for adequacy. If a documentation quality rating factor is desired with which to modify the quantity rating, a system of documentation rating may be used, as shown in Figure 9-15, which evaluates individual sections of the documentation. This review should always be performed by the same person. Each section of the program is rated, and the point value assigned is multiplied by the weight of the specific section. The total of the weighted values, divided by the maximum obtainable value, is the effectiveness rating. In order to retain optimum standards, a rule may be established that all sections rated below "fair" must be redone and the time for redoing included in the evaluation.

Testing Adequacy

A third factor affecting program quality is the adequacy of testing. The best measurement of testing adequacy is the number of errors found in the program after it has been turned over to production. Since this number is somewhat affected by running frequency (an annual program will not have very many errors turn up in the first year, a daily program may have many more) it should be put on a per run basis. Thus, if in a 10-week period, three errors are uncovered in a weekly program, the number of errors per run is .3.

Others

Three other factors relate to the quality of the program and the programmer. These factors have been discussed before; they are the comparison against standard of:

PROGRAM NUMBER _____ COMPLEXITY RATING [ ][ ][ ]

PROGRAM NAME _____ STANDARD NUMBER OF PAGES: _____

APPLICATION _____ *ACTUAL PAGE COUNT _____

SECTION	NUMBER OF PAGES *		EXCELL. 4	GOOD 3	FAIR 2	POOR 1	WEIGHT X	TOTAL VALUE	MAXIMUM VALUE	ACCEPTABLE
GENERAL DESCRIPTION	*						2		8	
FLOW CHART							1		4	
DETAILED DESCRIPTION	*						4		16	
MACRO-LOGIC CHARTS							2		8	
MICRO-LOGIC CHARTS							10		40	
SET UP INSTRUCTIONS	*						3		12	
OPERATOR'S INSTRUCTIONS	*						3		12	
PROGRAM HALTS	*						6		24	
FEATURES, OPTIONS, ETC.	*						5		20	
SPECIAL LISTS	*						3		12	
LAYOUTS - TAPE							2		8	
- CARD							1		4	
- PRINTER							1		4	
- MEMORY							1		4	
LISTING COMMENTS							4		16	
OTHER SECTIONS	*						2		8	
TOTALS							50		200	

*ADDED PAGES [ ] PER CENT OF MAXIMUM: [ ]

BY _____ DATE _____

Fig. 9-15. Rating of Documentation.

- Test shots
- Number of compiles
- Number of pages of documentation

All of these factors can be taken into consideration in evaluating effectiveness of a programmer at somewhat lower weight than the previous factors. Figure 9-16 illustrates the manner in which one company evaluates a programmer after completion and operation of his program. All of the above factors are included to come up with a composite score.

This composite score is the best possible manner in which to evaluate and compare all of the programmers in an installation.

All of the standards discussed have been rated through a combination of subjective and mechanistic measures. By using the same person as a rater, the measurement is objective in showing *relative* performance. It does not, however, provide for an *absolute* evaluation of the staff. If all of the programmers in a group are poor, and some are worse than others, the individual differences among them will be obtained in this manner; unless similar standard measurements of another installation are available for comparison, the fact that all of the programmers are sub-standard will not be discovered. To avoid this it may be possible to obtain an outside programmer, from another installation or from the manufacturer, for short periods of time to act as a control in establishing the

PROGRAMMER NAME _____ PROGRAMS COMPLETED:
RATING BY _____ A. _____
DATE _____ B. _____
C. _____

CALCULATION PROG. No.:	A.	B.	C. Example
A. STANDARD DAYS FOR THIS PROGRAM ARE			24½
B. ACTUAL DAYS FOR THIS PROGRAM ARE			22
C. PERFORMANCE EFFICIENCY A÷B			111
D. PROGRAM OPERATING EFFICIENCY			97
E. NET PERFORMANCE: $\frac{C+D}{2}$			104
F. NUMBER OF ERRORS FOUND – REVIEW AND OPERATION:			2
G. ADJUSTED EFFICIENCY: E – 2F			100
H. DOCUMENTATION QUALITY PERCENTAGE			86
I. ADJUSTED EFFECTIVENESS: $\frac{(H+20)+3G}{4}$			101.5
J. COMPILATION PERCENTAGE			108
K. TEST SHOTS PERCENTAGE			93
L. PAGES OF DOCUMENTATION PERCENTAGE			102
M. NET EFFICIENCY $\frac{(J+K+L)+7I}{10}$			101.2

Fig. 9-16. Programmer Evaluation.

absolute relationships. This is almost always necessary when using a less mechanistic method such as proficiency testing.

OTHER TECHNIQUES FOR PERFORMANCE MEASUREMENT

A number of other techniques have been used to measure performance. None of these techniques has been found to be as effective as that described here, but they are preferred by some because they are easier to install and simpler to operate. Brief descriptions of methods for the establishment of standards and comparative rating of programmers follow:

Maintenance of Historical Records

The simplest and most economical of all techniques is to develop detailed historical records about past performance. Nevertheless in many installations even the simplest records of performance have not been maintained and all of the valuable experience data has been irretrievably lost.

A historical record is set up for each program as it is assigned. The programmer is generally given responsibility for accumulating all required statistics, by task and program segment. A basic time record appears as Figure 9-17.

In addition, the programmer would record such facts as:

- Number of test shots
- Number of compiles
- Total machine time used
- Number of pages of block diagrams
- Number of pages of documentation

This information would be used for program cost analysis, but more important, would be filed with the program for future reference. If another job is required of a similar nature, or requiring segments similar to those of a completed program, the historical record can be used as a

	TIME IN DAYS:						
PROGRAM SEGMENT	MACRO	MICRO	CODING	DESK CHECK	TEST	DOCUMENT	TOTALS
SEGMENT 1							
SEGMENT 2							
SEGMENT 3							
TOTALS							

Fig. 9-17. Basic Time Record.

guide in estimating the cost of the new assignment. There are several advantages to this method: the cost is low, the amount of record keeping is small, and there is no cost of after-the-fact evaluation. Disadvantages, which probably outweigh the advantages, include these:

- There is no comparative evaluation by programmer
- It is difficult to get an estimate of final completion
- The basis for comparison is the performance of someone whose efficiency or training is not known.

Nonetheless, if no other records are maintained, there is a considerable amount of value in recording vital experience data.

Personnel Evaluation: Proficiency Testing

Two basic personnel evaluation techniques can, when combined with historical records of performance, provide some insight into staff efficiency and aid in estimating performance. The first is a proficiency testing technique, used in several installations to determine the relative proficiency of the programming personnel. The basic test is assigning the same programming problem to all members of the staff at the same time. This can be a small subroutine or a moderate size program, depending on the time available.

Each programmer is given a detailed specification of what is to be accomplished by the routine. Each is told what he is supposed to produce, i.e., documentation, block diagram, flowcharts and the like. If it is a simple problem, it may be completed within one day; if it is more extensive, the programmer time should be carefully logged on each of the tasks performed. The factors measured include:

- Elapsed time by task
- Number of instructions used or memory space used
- Running time of the routine on sample data
- Number of errors uncovered
- Amount of machine time used
- Quality of documentation

Each of these factors is weighted and the score is totalled for each person. For control purposes one or more "outsiders" should be invited to sit in on the tests. These may be obtained from another installation, from the manufacturer, or from a consultant. Some indication should be obtained of the quality of the outside persons, to enable some judgment as to the ranking of the staff.

Personnel Evaluation: Ranking Tests

A second method of personnel evaluation is used for relative proficiency ranking. This technique must be used with extreme caution and a detailed explanation to the staff, since it may have an adverse effect upon morale. The basic technique is to ask each person, by questionnaire, who he considers to be better than himself, worse than himself, etc. Typical questions could be:

a. Who do you think is the best programmer in the installation?
b. Who do you think is the worst programmer in the installation?
c. Who, in your opinion, is best at testing? at documentation? etc.
d. If you required technical assistance, to whom would you go?
e. To whom would you never go for help?

On the basis of these questionnaires, which can remain anonymous, a ranking can be established of all programmers.

Personnel Evaluation: Ranking Tests

A simple measurement standard, a direct function of program size alone, is time per program instruction. A common standard is one instruction per hour, taking into account definition, macro, micro, coding, test, and documentation. A program requiring 250 instructions, would require 250 hours to complete. Assuming an 8 hour day, this would be 31¼ man days. Another way of stating this standard is as cost per instruction.

An installation with 10 programmers, at an average annual cost of $10,000 including all fringe benefits, producing 20,000 instructions in a year has an average cost per instruction of $5. Since each of 10 programmers works 241 days and 8 hours per day, the total number of hours is 19,280. It therefore requires approximately 1 hour for each finished instruction.

STANDARDS FOR OPERATING PERSONNEL

The major tasks of operating personnel are:

- Program set-up
- Console operation
- Housekeeping
- Log record keeping
- Program take down
- Emergency handling (less frequent)

Performance standards can be applied to set-up and take-down; the other functions are overlapped with machine operation, so that their efficiency is of limited concern. Standards for set-up and take-down have already been discussed in Chapter VIII, so that little remains to be said about specific performance measurement for operating personnel.

Another technique for establishing operating performance standards is the application of time and motion study to the set-up and take down functions.

PERFORMANCE STANDARDS FOR SYSTEMS ANALYSIS

The systems analysis functions are not as well defined as those of programming. Nevertheless, the same approach to personnel performance standards can certainly be used. The techniques described below are experimental in that they have not been validated by extensive field experience as have the techniques for programming performance. They are presented here to show a logical approach to the problem.

Systems Analysis Parameters

Several parameters can be defined as critical in systems analysis. These appear to be:

Complexity.—As in programming, one of the most critical parameters, with a similar direct relationship to performance time, is complexity. To avoid confusion the same definition and scale of complexity as evolved for programming can be used, i.e.:

A Simple
B Moderate
C Difficult
D Complex
E Very Complex
F Impossible—out of range

The size of a systems analysis assignment is not easily stated. Special factors must be developed to describe required analysis time.

Number of Documents.—The number of documents currently produced, or to be produced influences time required, since it represents the number of *different* reports that the analyst must analyze or design.

Number of Functions.—The number of manual functions *currently* performed to produce the system output is another important time factor. The number of steps to be included in the *new* process is another. The total steps to be performed is the sum of the old steps and the

new steps, in effect the number of symbols to be drawn on the flowcharts of both the new and the old system.

Tasks of Systems Analysis

Although systems analysis tasks are different under different circumstances, it is necessary to establish a fairly formal set of tasks if a similar approach to that used for programming is desired. System analysis tasks vary according to

- Whether the existing system is a manual system or on punched cards
- Whether the new system is totally new to the company
- The amount of redesign desired

It is nevertheless possible to define common tasks, and if they are sufficiently detailed, the estimator who develops specific standards for each job can eliminate those which are not to be performed. These tasks are:

- Assembly of available data
- Initial interviews of persons involved in current system
- Additional interviews of others involved in current system
- Development of present system flowcharts
- Analysis of present system documents
- Analysis of present system files
- Flowcharts of the new system
- Document design or re-design
- General description of both systems
- Layouts
- Timing
- Job specification completion

Relationships between Tasks and Time

The following are experimental relationships which have been developed to illustrate the approach and provide an example for use in the construction of specific, meaningful standards.

Assembly of Available Data

		Man-days for level of complexity	
		½	A
		½	B
1 hour for each document value	*plus*	1	C
		1½	D
		2	E

Initial Interviews for Each Function

		Hours for level of complexity	
		0	A
		½	B
½ hour for each function code	*plus*	1	C
		1½	D
		2	E

Interviews of Additional Persons per Function

Interview time for each person on each function equals ½ hour

Development of Present System and New System Flowcharts

		Man-days for level of complexity	
		0	A
		½	B
¼ hour per function	*plus*	1½	C
		2½	D
		3½	E

Document Analysis: Present System and New System

		Man-days for level of complexity	
		0	A
		1	B
1 hour for each document	*plus*	2	C
		3	D
		4	E

Analysis of File Retention

2 hours for each file

Document Design

2 hours for each document to be designed or changed

General Description, Layouts, Timing and Job Specification

Man-days for level of complexity	
2½	A
3½	B
5	C
7	D
8½	E

Summary for Average Systems Design

Complexity:	C
Number of functions present system:	32
Number of steps in the new system:	14
Number of documents to be changed:	16
Number of new designs:	2
Number of interviews per function:	2
Number of files retained:	4

Function	Man-Days	Hours (converted)
Assembly of Data	2.6	
Initial interviews	6.0	48
Additional interview	2.0	16
Flowcharts of present system	2.5	
Document Analysis	4.0	
File retention	1.0	
Flowcharts of New System	2.0	
Document Design	.5	
Job Specification	5.0	
Total	25.6 man-days	

A simplified approach to this problem, developed some years ago by The Diebold Group, Inc., is illustrated as Figure 9-18. In this instance, the complexity of the application was not included in the analysis, and only the functions and the document designs have been included in the time evaluation. Nonetheless, a chart such as has been illustrated could be used easily as a simplified estimate of time.

AREA PERFORMANCE BUDGET
PLANNING AND DEVELOPMENT

A. *Operations Analysis*

a. No. of functions or steps	b. Standard time per initial person on function	c. a × b	d. Adjusted No. of occurrences of same function*	e. Std. time for incremental person	f. d × e	g. Total Time c + f
40	90 min.	3600	23	20 min.	460	4060 min.

* The adjustment involves eliminating the "original" person on each function.
The figure here represents the total occurrences of such a situation, rather than no. of people. If a total of two people each do three different functions, then the number here would be 3 or $(2 \times 3) - 3 = 3$.

B. *Inputs, Outputs and Files*

	h. Standard time	i. Number of occurrences	j. h × i	k. (j totaled)
Input form	30	2	60	
Output form	40	6	240	
File Reference	10	7	70	370

C. *Total Time—System Study*

Column g	4060
Column k	370
	4430 man minutes

The same form is used as estimates and recapitulation. For the recapitulation, the numbers in columns a, d, and j, are actual rather than estimated. Similar forms, with column titles changes as necessary are used for system design and programming.

Fig. 9-18. Performance Evaluation: Systems Analysis. (*Courtesy*, The Diebold Group, Inc.; © 1958, ADP Co., Inc.)

CHAPTER SUMMARY

The parameters explained in Chapter VIII for the definition of programs are used to develop time relationships for the tasks involved in programming. These tasks include the initial analysis, the development of the macro- and micro-block diagrams, and the coding, testing and program documentation.

After the necessary performance standards for programming have been established, a schedule is developed. The schedule assists in the evaluation of performance data, gathered in the same detail with which the standard was established. The data evaluation provides management with accurate information on programmer performance and the progress of work and helps in the specialization of personnel by capability and the measurement of personnel training time and effectiveness.

Management can now take action based on realistic standards and timely performance information. The same methods of reporting and evaluation can be used to assess the effects of management action, and to adjust it if necessary. This is the basic definition of management control.

Equitable and useful personnel performance evaluation requires that measures of quality be established. Performance efficiency ratings must be adjusted by quality measurement.

Other techniques for performance evaluation are briefly explored. They include the use of historical records, and the evaluation of personnel on a ranking or direct proficiency basis. The Chapter closes with a brief analysis of a possible approach, parameters, and formulas for systems analysis standards.

Questions for Review

1. Considering the parameters selected for programming, indicate others which you might use for programming and systems analysis.
2. Design a simple system to process a weekly payroll, and retain the necessary records for quarterly and end-of-year processing. Rate the programs which you have developed, select the machine, and establish the performance measurement standards which apply.
3. Indicate measures of quality which you would apply to the programs.
4. Why is it so important to have only one person rating quality and other subjective measures?
5. Develop a progress reporting system for the programmer(s) which you will assign to the payroll problem.

6. Establish evaluative techniques to provide management with information required to effectively manage the development program.
7. Assume that the following information has been returned to you about the payroll programs designed in Question 2:

Program Number	% Complete	Reported Time						
		T 1	T 2	T 3	T 4	T 5	T 6	Total Days
P01W	100	2	4	6	8	6	4	30, Programmer *A*
P02W	50	1½	5	7	2			15½, Programmer *B*
P03A	75	4	6	12	4	2		28, Programmer *A*
P04W	100	1	4	6	3	11		25, Programmer *B*

Develop relationships between programmer *A* and *B;* establish their individual ratings, and the tasks at which each appears most proficient. When will the total job be completed, at current efficiency?

8. If the tasks needed to be completed one week sooner than the current schedule above indicated, what action would you recommend to management to speed it to an earlier completion?

Chapter X

OTHER USES OF PERFORMANCE STANDARDS

INTRODUCTION

Management can derive additional benefits from performance standards. These benefits are achieved through evaluation of the information supplied. The use of standards falls into several categories, namely:

- In the *personnel program,* standards are used to gauge training effectiveness and speed, and to obtain and evaluate experienced personnel from external sources.
- In the development *schedule,* standards are used to provide management information necessary to alter progress, plan and project future actions, and determine phase completion dates.
- In the *projection* of future needs, specifically in assessing the effect of management decisions upon data processing requirements and the actions needed.
- In establishing a realistic *budget,* which cannot be done meaningfully without performance standards.
- In *estimating costs* of changes to a system, new development, and continued operation.
- In proper *cost accounting,* development and operating functions.

These topics are discussed in greater detail in this Chapter.

THE PERSONNEL STAFFING PROGRAM

The personnel program established for computer installation development, or for maintaining an operating installation, consists of the following elements:

- Determination of personnel sources
- Methods of recruitment and selection
- Training and evaluation.

Performance standards are used in the latter two, to evaluate new personnel and to establish a meaningful learning curve. Some consideration should be given to the sources of available personnel, and the *methods* standards to be applied to the hiring and selection process.

Personnel Sources

The first major decision to be made by management is whether to promote from within or to go outside for experienced personnel. If inexperienced personnel are suitable, there is little excuse for not promoting from within; only when experience is desirable should external sources be considered. In either case the selection process is the same, except that experienced outside personnel should be evaluated against a performance standard in order to determine the value of their previous experience.

Personnel obtained from within come from a number of sources. This fact is strongly brought out in a study of 20 large corporations performed by the Bureau of Labor Statistics.* A summary of this study is shown in Appendix C, Table C-1.

Outside sources are also varied. Advertising may be used to attract experienced programmers at an approximate cost of $250 per programmer hired. Similarly, personnel agencies perform a screening function (sometimes extremely valuable), at an average cost of $400 per programmer. College recruitment is not satisfactory in that very few experienced programmers can be obtained in this manner; if trainees are desired this method is valuable, provided that all eligible employees have been given first consideration.

Standards for Selection of Trainees

Table C-2 of Appendix C shows a distribution of traits considered desirable by corporations interviewed in the Bureau of Labor Statistics study. To determine possession of these traits and the general suitability of the trainee, the following procedure is suggested.

1. Determination of Interests, Hobbies and General Attitudes.—An interest questionnaire is suggested. This questionnaire should pinpoint the basic interests, hobbies and external sources of relaxation. This will

* "Adjustments to the Introduction of Office Automation," *Bulletin* No. 1276, Bureau of Labor Statistics, 1960.

enable an evaluation of the candidate's motivation and personal circumstances.

2. *Determination of Aptitudes.*—A conventional aptitude test may be used. This can be obtained from the manufacturer, or any qualified organization specializing in the selection of data processing personnel. Available tests include general machine operator aptitude tests, programmer aptitude tests, and tests for the evaluation of analysts. Some caution should be used in the evaluation of test results; correlation has been satisfactory, but success in the tests does not guarantee success in the job. Similarly, a failure in the tests should not alone disqualify the applicant, if other characteristics point to strength undetected by the tests. Older applicants will generally have more difficulty with the tests, because they have been removed much longer from an environment in which competitive tests are administered.

3. *Determination of Motivation.*—The only satisfactory method of establishing the motivation of a candidate, extremely important to his productivity potential, is a personal interview. This requires considerable experience in personnel work rather than a deep understanding of data processing functions.

Standards for Selection of Experienced Personnel

If experienced personnel must be obtained outside of the organization, the selection process will include the procedures commonly used for professional personnel. This includes the job application and the interview; it may also include psychological tests. The basic programming aptitude test should also be used, but it is entirely possible that the same or a similar test has been given to the experienced programmer before.

Of major importance in the hiring of experienced personnel is the testing necessary to determine the value and extent of experience. These tests often pinpoint both the quality of prior experience, and the productivity level the applicant has achieved.

The test design should cover:

- General knowledge of equipment characteristics, programming languages and techniques, and the like
- Accuracy and attention to detail
- Logical ability
- Performance as measured against the standard.

The test should include a series of questions about the equipment as well as small combinations of instructions in error which the applicant may be asked to correct.

Another part of the test should consist of an established subroutine or problem full of errors which the candidate should be asked to correct and identify. The candidate should be told the exact number of errors in advance, so that he will continue to work on the problem until it is error-free.

Also included should be a small incomplete block diagram with the basic linkages left open. To complete the diagram, the candidate must possess an understanding of block diagramming and of logical analysis. All parts should be time limited.

The final test should be the programming of a complete subroutine or small program, which normally should be accomplished in 3 to 4 hours. Required should be a block diagram, the requisite coding, and a brief statement of necessary documentation, or the documentation itself in simplified form. Scoring should measure time used against standard quantity and program quality, measured by neatness, accuracy and logical completeness.

Standards for Personnel Training

A regular training program should be established for new data processing personnel, especially in programming and systems analysis. The program should include:

1. *Formal Schooling.*—The equipment manufacturer operates training schools for computer operators, programmers, and systems analysts. In most cases these schools should be used to impart the rudimentary knowledge necessary for performance in one of these positions.

2. *On-the-Job Training.*—Six to 9 months are generally required for a trainee programmer to become a productive worker. During this critical period, the work assignments should be carefully organized under the direct supervision of a senior programmer who is also a capable instructor. The stages of on-the-job training should follow this rough sequence:

   a. Assignment of a simple problem to be completed under the control of the senior programmer (this may require 4 to 6 weeks). The problem is used strictly for training and the output is discarded.

   b. Assign the programmer-trainee to assist a programmer or senior programmer in the simpler tasks of a usable problem. The first task assigned usually is documentation; the second is part or all of the coding.

   c. After the trainee has become proficient at the simpler tasks (75% of standard), assign a simple problem to be completed in its entirety under continued supervision.

d. Assign a more complex problem until an acceptable percentage of standard has been reached.

3. Continued Education.—A program of continued professional development should be initiated. This might take the form of weekly sessions in techniques, new approaches, and the like. Programmer trainees should be exposed to sessions on technique. Senior programmers should be exposed to sessions on systems analysis, operations research, and other advanced disciplines.

Evaluation of Training

Programmers are often classified exclusively by tenure. It is common to see a personnel qualification such as the following:

1 to 6 months—Programmer Trainee
7 to 18 months—Junior Programmer
18 to 36 months—Programmer
Over 36 months—Senior Programmer

This classification fails to take individual differences into account. One programmer may require a longer time to become qualified because his learning rate is lower than average; another may be extremely adept and become a qualified programmer or senior programmer in a short period of time. A better approach to classification would be to use performance standards as a basis in qualifying the staff, for example:

Programmer Trainee.—An employee will remain a trainee until he has the ability to do productive

Coding at 75% of standard (90% quality)
Testing at 60% of standard (90% quality)
Documentation at 80% of standard (80% quality).

His classification may not be changed in the first three months.

Junior Programmer.—An employee will remain a junior programmer until he has achieved the following:

Block diagramming at 65% of standard
Coding at 100% of standard
Testing at 95% of standard
Documentation at 105% of standard

No classification change will occur in less than 6 months.

Programmer.—An employee will remain a programmer until he has achieved the following standards:

Macro-logic at 100% of standard
Micro-logic at 110% of standard
Coding at 120% of standard
Testing at 100% of standard
Documentation at 120% of standard

The overall quality rating of a *programmer* must be at least 100% before promotion to Senior. No classification change will occur in less than 12 months.

On the basis of the above rules, the best trainee would not be able to achieve Senior Programmer in less than 21 months. The Senior Programmer classification should be restricted to individuals who consistently exceed standard.

Similar rules could be developed for operators, whose classification would probably include only

Operator trainee
Operator (sometimes called peripheral operator or tape handler)
Lead operator (or console operator).

Promotion depends upon an evaluation of performance against the standards and also a review of the attitude and interest required in a really good operator.

Continued Personnel Evaluation

After training, the most obvious use of performance standards is evaluation of the performance of each staff member, as explained in Chapter IX. Another use of personnel performance standards, indicated in Chapter IX, is as an aid in functional specialization of the staff.

The continued maintenance of a learning curve, also illustrated in Chapter IX, provides an indication of the learning trend: i.e., a positive slope indicates continued improvement, a straight line indicates maximum absorption in the present job, and a negative slope indicates a change of interest or attitude. This is a management tool which should be maintained for each programmer.

THE DEVELOPMENT SCHEDULE

Accurate performance standards are obviously necessary to the establishment of a realistic development schedule. By developing the systems flowcharts, and by rating the programs of which it is composed, the detailed development time can be readily determined.

The schedule normally set up should be segregated by programmer.

Each programmer should be assigned to a series of programs, by himself or as a member of a team.

The assignment of programs is usually based on the experience and competence of the programmer, the application area in which he has experience or interest, and the complexity of the program. The detailed schedule should generally be given to the programmer. In cases where programmers are consistently unable to meet the established deadlines, it is usually better not to give out the estimated completion date; it will only discourage the less able staff members.

The overall schedule completion date can be readily determined at the outset of the development program. This date should be regarded as tentative, because of its assumption that all members of the staff will work at 100% of standard. As soon as some experience has been obtained, however, actual efficiencies can be applied to the schedule as illustrated in Chapter IX. This permits recomputation of the completion date, taking into consideration the effectiveness ratio of the staff in each function.

The completion date should continually be checked to insure that no major changes in efficiency have occurred. Once it has been firmly established, it can be used to plan for equipment delivery, site preparation, which may require a six month lead time, and the training of operators and other personnel.

Figure 10-1 is part of a typical programming schedule, developed on the basis of established performance standards. Buffer areas are included within the schedule, to allow for illness, personal time off, or other possible delays.

PROJECTION OF FUTURE NEEDS

Because of the long lead time required to develop a program, it is important to determine rapidly the future use of data processing manpower and machines. All data processing resources must be scheduled well in advance and a method must be available to estimate quickly future requirements.

Current equipment utilization, based upon the standards developed in Chapter VIII, can be measured and estimated accurately. Using the same standards, volume projections can be made for the future, and costs, which increase correspondingly, can be estimated. If volume is estimated to increase by 40% and this requires a 30% increase in data processing productive time, the standard ratios for rerun time, unscheduled maintenance, and the like, can be applied to arrive at total workload. If no new applications are developed in the period under consideration, the estimated set up time will remain fairly constant, and the testing and assembly time strictly proportional to the percentage of changes made.

If projections indicate that the machine will no longer be adequate, a long range plan must be developed for meeting this condition. At this point there are several alternate routes:

- Changes in operating procedures to permit existing equipment to handle the increase
- Adding a smaller machine to handle overflow, sorting, and the like
- Adding a second identical computer
- Changing to a larger computer.

Exacting performance standards permit direct estimation of the costs which are involved in each alternative. Personnel costs of reprogramming, or of converting certain programs to a smaller system, can be estimated in detail. Another decision factor may be timing. Only with adequate performance standards and knowledge of operating efficiency, can a reliable estimate be made of the completion date of alternative courses of action, and this can show the latest point at which a choice among them can be made.

PROGRAMMER	MONTH 11				MONTH 12						MONTH 13				
PROGRAMMER A	PROGRAM: P01W C 6/5				T3	B	P03D T1	P03D T2	P04D T4		T5	P05D T1	P05D T2		
	T1	T2													
PROGRAMMER B	PROGRAM P01Q												P02W		P03M
	T1	T2			T3						B	T4			T4
PROGRAMMER C	P01D					P02D									
	T5				T6	B	T1				T2	T3			
PROGRAMMER D	P04M	P02Q	VACATION		P02Q						P01M				
	T3	T4	VACATION		T5		T6				T1	T2			
KEYPUNCHING	P02Q	P04M				P01W						P01Q			

Fig. 10-1. Programming Schedule for Payroll Application.

ESTIMATING

Data processing management is responsible to the user of its services to accurately estimate costs of the job before undertaking the assignment. In those companies where the data processing function is established as a service it should be asked to *bid* on the performance of work by accurately estimating operating and development costs.

In many cases, operating management has no idea of the costs involved in data processing. It is not at all uncommon for an operating manager to request a report, or a change in a report, without any understanding of the costs of such a request. The data processing department, not realizing why the request is made, makes the change regardless of its cost.

This has created a number of problems in communications and in relations between operating departments and data processing. A procedure should be set up which forces the data processing group to estimate the cost of the request, and to submit this cost to the user for prior approval. This will enable the operating manager to judge whether or not the need warrants the cost.

In order to establish such a procedure, it is necessary to have accurate performance standards. Without standards, the programmer is responsible for estimating his own time on the job. If the request is for a change to an existing program, the programmer will usually overestimate the cost. If the system is currently operating on the computer the programmer may overestimate the cost in order to force the user away from making the change. If the user goes ahead anyway, the extra time easily can be used by the programmer and charged to jobs that were underestimated.

When performance standards are uniformly administered by one person, these attitudes cannot affect estimating of development time and operating cost. The user is given an honest representation of the costs, and can fairly estimate his own need against the cost to the company.

Figure 10-2*a* illustrates a form used to estimate the cost of a data processing job from original systems analysis to documentation. The reverse side of this form is shown as Figure 10-2*b*, and is used to estimate operating costs.

Comparative Cost Estimating

An interesting use of personnel performance standards is for establishing comparative costs of programming and development for different machines or different programming languages. The latter was illustrated

DATE | **DATA PROCESSING JOB COST ESTIMATE** | BY:

LOG NO. | TITLE: | REQ : DEPT :

DESCRIPTION		TASK CODE	ANALYSIS SCHEDULE DUE DATE	ANALYSIS HOURS
LIAISON	CONFERENCE	01		
	SYSTEMS	02		
FORMS	CARD	11		
	REPORT	12		
	SOURCE	13		
PROCEDURE	CLERICAL	21		
	KEYPUNCH	22		
	OPERATION	23		
PROGRAM	FLOW CHART	31		
	WRITE TEST	32		
CONTROL PANELS	DIAGRAM	41		
	WIRE-TEST	42		
PREPARE	ESTIMATE	51		
	TASK BROCHURE	52		
REWORK	CHANGED REQUIREMENT	61		
			TOTAL	

OPERATIONS REVIEW

RECOMMENDATION — ANALYST-PROGRAMMER

SPECIALIST

ANALYST SUPERVISOR

EQUIPMENT SUMMARY				
TYPE	HOURS	COST	PANELS	COST
TOTAL				

LABOR SUMMARY				
CLASSIFICATION	SET UP HOURS	SET UP COST	RECURRING HOURS	RECURRING COST
ANALYST				
CLERICAL				
KEY PUNCH				
OPERATOR				
PROGRAMMER				
TOTAL				

MATERIAL SUMMARY		
	SET UP	RECURRING
CARDS		
FORMS		
PANELS-WIRES		
TOTAL		

BUDGET SUMMARY		
	SET UP	RECURRING
MATERIAL		
LABOR		
EQUIPMENT		
TOTAL		

JOB NO.

FORM A1938 (2-60) AS

Fig. 10-2*a*. Data Processing Job Cost Estimate. (*Courtesy*, General Dynamics/Astronautics Division)

in Chapter IX, where different standards were developed for programs written in Autocoder III and COBOL, and for programs written in FAP and FORTRAN. The reduction of programming costs is clearly demonstrated by these examples. The FORTRAN program took 43 days; if it were written in FAP it would take 59½ days. A COBOL program which could be completed in 47 days would take 56 days in Autocoder. By rating every program to be written, and calculating the cost of programming using the performance standard as the base time, a cost comparison can be made of COBOL vs. Symbolic, or any other two languages. Lower programming cost and time must then be weighed against possible loss in program efficiency. (Appendix B shows approximate losses in efficiency against the cost saving for the higher level languages).

A comparative cost estimate can also be made for programming of two different machines. Using separate standards, and designing the basic system separately for the characteristics of each machine, it is quite feasible to arrive at independent programming cost estimates. This comparison is very useful in the equipment selection procedure.

DATE: **DATA PROCESSING JOB COST ANALYSIS-OPERATIONS** BY:

LOG #: TITLE: REQ. DEPT.

OPERATION	MACH. TYPE	INITIAL SET UP					RECURRING:				
		CARD VOLUME	OPER. HRS.	OPER. COST	MACH. HRS.	MACH. COST	CARD VOLUME	OPER. HRS.	OPER. COST	MACH. HRS.	MACH. COST
TOTAL											

FORMS REQD.	TYPE	SET-UP	RECUR.	ANNUAL	SCHEDULE
		$	$	$	
TOTAL COST		$	$	$	

Fig. 10-2*b*. Data Processing Operations Cost Estimate. (*Courtesy*, General Dynamics/Astronautics Division)

Another useful function for which performance standards may be used is estimating the cost of conversion programming. No matter what conversion method is used, the tasks to be performed are measurable in relation to program size, complexity, and input-output requirements. If, for example, the block diagrams are current and usable in the new system, the only necessary tasks are coding, desk checking, testing, and change of existing documentation.

A similar use of performance standards is for evaluating different methods of performing a programming task. If, for example, a given system is to be developed using contract programming, the cost of this programming can be compared to the cost of doing it "in-house." If other methods of performing the task are possible, the same kind of comparison can show which is cheapest and fastest.

BUDGET

A typical operating budget for a data processing department might be set up as follows:

Object	*Budget Last Period*	*Actual Last Period*	*Budget This Period*
Supervision			
Operating Personnel			
Programming Personnel			
Clerical Personnel			
Equipment Rental			
Peripheral Equipment-Punched Card			
Personnel Fringe Benefits			
Supplies			
Overtime			

The budget should also include allocations for space charges, light, heat, power, telephone, postage, etc.

The major budget items that can be established by using performance standards as the basic guide are:

- Personnel overtime
- Equipment overtime
- Programming and clerical personnel.

In every one of the expense objects listed above except equipment operation, cost is a function of the amount of work to be performed. Equipment operators and base equipment rental do not vary with workload; the basic cost must be paid regardless of the operating volume.

Overtime costs can be estimated by determining the standard production volume achievable in a basic shift operation, e.g., 10,000 employee payroll, 200,000 inventory items with 5% average activity, 40,000 customer accounts updated daily, etc.

The curve on a graph of production capacity against projected volume should indicate whether overtime or a second operating shift is needed, depending on the amount of additional time needed.

Another method of budget estimating is to use a linear relationship of costs to volume. Although this is not entirely accurate it is fairly conservative. Operating costs of the machine are basically linear with volume, although this depends on the slope of the cost/volume line (as illustrated

in Chapter VIII). An increase in volume will not, however, increase the cost of set-up, nor will it affect the cost of testing and assembly in a linear manner. These costs must always be estimated separately.

The programming personnel cost included in the budget consists of two functions: program development and maintenance. The latter is, of course, a function of the number of operational programs and the frequency and complexity of changes. The former depends on the number and difficulty of new applications required. However, available machine time, which depends on volume, will affect the amount of new development work.

The development of a realistic operating budget involves the following steps:

1. Estimate the volume of business in each current program and in new programs scheduled for operational status.
2. Calculate machine time required: productive time as a function of volume, set-up time as a function of the number of programs, rerun time as a standard percentage of productive time, and so on.
3. Calculate the basic cost of machine rental and overtime required. Calculate the overtime cost of personnel or, if necessary, the cost of an extra shift.
4. Calculate the number of maintenance programmers on the basis of the number of changes *expected.* Assuming an average size and comparatively few changes, estimate the approximate standard cost per change, and so derive the number of maintenance programmers required.
5. Determine if sufficient machine capacity is available for the inclusion of desired new applications. Estimate the programming cost of the new applications according to the standards and using standard costs, distributing these costs over the time required. Based on this, estimate the number and cost of the new development programmers required.
6. According to the estimated size of the staff, estimate the number and cost of the supervisory personnel and the number and cost of clerical support personnel.
7. The cost of supplies, fringe benefits, and the like can be calculated as a percentage of labor and machine costs.

Budgeting a New Development Program

Techniques used to established a budget for a new development or installation program are quite similar to those given, to the extent that performance standards are used to estimate the costs required. The cost categories are:

- Supervision
- Systems analysis staff
- Programming staff
- Personnel recruitment and training
- Supplies
- Space charges
- Fringe benefits
- Site preparation, air conditioning, decorating, etc.
- Travel and other testing costs (possibly including machine time)
- Conversion: machine time, supplies, unit record preparation, etc.
- Parallel operation of a dual system
- Consultants, architects, engineers
- Magnetic tapes
- Computer room hardware, including equipment used to set up tape library, carts, shelves, fireproof vaults, etc.

Normal budgeting methods can be used to estimate the cost of supplies, space charges, magnetic tapes, and outside consultants. The cost of site preparation results from competitive bidding and the cost of the miscellaneous computer room hardware can be obtained from appropriate manufacturer's catalogues.

A basic system has already been designed for the "bread and butter" application as part of the feasibility study. It is therefore possible to use the techniques outlined in Chapter IX to establish the exact number of man-days required to develop the system, for systems analysis and for programming. This man-day estimate can easily be translated to man years: there are 241 man-days in a man year, if there are 7 holidays, 2 sick days, and 10 vacation days to reduce the 260 weekdays available.

The man year estimate indicates the work to be performed; elapsed time is a function of the number of men used. Twelve man years of programming can be done by 12 people in one year, by 6 people in 2 years, by 4 people in 3 years, or by 3 people in 4 years. The date when the major application must be finished depends on such factors as return on investment, rate of business volume increase, and other factors not necessarily known to data processing management. After top management, knowing the cost, specifies that the work will be done over a given period, the number of programmers and their annual cost is easily derived. The cost of supervision varies with the number of programmers—an average of one lead programmer, or group supervisor, is required for administration and guidance of 12 programmers, in addition to the Data Processing Manager.

The remainder of the budget requires little calculation. The cost of hiring is proportionate to the number of people required, the immedi-

acy of the requirement, and the experience level. The cost of training may vary from six months for an apprentice to one month for an experienced programmer new to the organization.

Fringe benefits costs are, of course, directly proportional to the annual payroll, as are clerical costs directly attributable to supervision and documentation. Travel and miscellaneous costs for testing depend on the location of the nearest available compatible computer system. If it is within the same city, the travel expense will be limited; otherwise, travel costs may become substantial; after each programmer has obtained some machine experience, remote testing should be instituted to reduce both travel expense and the direct testing costs.

If machine time for testing is charged, an estimate must be made. Again, performance standards will be used, since a direct count of the number of required test shots should be available. By multiplying the number of machine shots by average machine time (including set up if this is charged for), the total machine time requirement can be obtained easily. Even if some of the testing is done after the equipment has been delivered, a cost is attached to its use, and it should be appropriately budgeted.

Conversion cost may be quite heavy. Included is the cost of writing data conversion programs, unless the system programs provide for such conversion. If present data is in card form, the conversion may only involve reformatting or the operation of a card-to-tape conversion program. If some of the data is available in card form and other data is new, the new data must be coded and punched into cards. If the current system is completely manual, all of the data must be punched into cards or tapes and converted into the desired format. Other conversion costs are training of affected personnel, training of operators, and the time not included in the standard required for programmers to assist in the conversion.

A conversion factor often ignored is validation of input data. If the data is currently available in card form or on paper and is to be converted to cards, its validity must be checked. Stray column punches creep into card data files and do not affect the card operation, but they may cause havoc in a computer. New information must be coded in a form not previously used, and this must be equally rigorously validated. Most programming logic assumes that the information on tape is valid and that no further checking need be done. Therefore all checks must be built into the conversion operation, or the logic of the daily work will fail completely.

Parallel operation costs must be separately estimated. This depends on the length of time the system will operate in parallel, which will vary with the accuracy of programming, the accuracy of conversion, and the

type of system from which conversion is made. It may be a minimum of one week and up to six months at a maximum. The budget for this operation is usually the cost of operating the computer systems; i.e., the cost *added* to the normal operation. A second factor, often ignored, is the cost of checking the output of both systems, and when one is in error tracing the error to insure that it is corrected properly (a parallel operation frequently uncovers major errors in the old system as well).

A new development program, regardless of its budget, can be capitalized and amortized over the period in which the savings are expected to occur. This reduces the adverse effect on earnings which a large scale development program may create.

COST ACCOUNTING

Distribution of the operating costs of a data processing operation has already been discussed in some detail in Chapter VIII. The methods given based charges on utilization, or on an estimated percentage distribution among users. In many instances it may be necessary to charge data processing costs as a direct operating expense, that is, as part of product costs.

In this case the two basic accounting methods available are job cost accounting (project or direct cost accounting) and standard cost accounting (indirect cost accounting). The first establishes a job number for each product or project and all labor and machine rentals are charged accordingly. This simple system is helpful in product profit-and-loss accounting. In data processing this method has some disadvantages; for example, the cost of rerun would be directly charged to the project whose work was being redone, even though the basic cause was machine or operator failure. Similarly, the costs of testing, set up, assembly, and the like would be charged directly, and operator or programmer inefficiency would be charged to the project on which it occurred.

Standard cost accounting avoids this kind of situation. Under this method, established performance standards are converted by the accounting department into standard job *costs,* on which basis charges are made, including all equipment rental and costs related thereto, operators, programmers, and all indirect charges.

Variances from standard job costs which actually occur represent gains or losses in efficiency. The variances should never be charged to the individual job account, since the job should not gain or lose on the basis of the performance of the data processing department. As a result, a separate variance account is established to post all charges over standard, and all credits for performance better than standard. If the net variance of the department is positive, a departmental profit and loss statement

should be enhanced. This method of standard cost accounting is both equitable to the using department, and represents a greater incentive to the departmental manager and his staff.

CHAPTER SUMMARY

This Chapter has further explored the uses that can be made of performance standards. Included is a brief discussion of the use of performance standards in establishing a good personnel program, both in personnel selection and training. It was first necessary to explain briefly selection and training techniques and to demonstrate methods for evaluating personnel during training, for salary review and promotion.

Other performance standard uses include the setting up of a development schedule and using the same concepts, forward management planning of all types. Performance standards are also used to estimate the cost of making changes and creating new applications. The same kind of cost estimating techniques can be used to compare machines, methods of programming and the use of various programming languages.

A natural outgrowth of the development of realistic standards is the ability to develop a meaningful budget. This in turn leads to the development of an equitable cost accounting system.

Questions for Review

1. Assuming that you have an established personnel program, relying heavily upon personnel performance standards in programming and machine operation. Develop a method for using these performance standards to enable direct incentive payments through:
   a. a system of payment for work produced (i.e. piecework programming).
   b. a system of paying an incentive bonus, based upon accomplishments as a function of the standards.
   c. a system for providing merit raises at periodic intervals to persons reaching certain levels of standard.
2. Develop the entrance requirements for experienced systems analysts. Develop the same requirements for trainees for that position. Do you feel that an analyst must have programming experience?
3. Develop a personnel budget for the development of a payroll program. Assume that you have the following staff:

1 Trainee	$ 92 per week
1 Programmer	$145 per week
1 Programmer	$160 per week
1 Senior Programmer	$207 per week

What is the "standard cost" which you would use? How would you accomplish this development? Would you be able to do it more economically on a different basis, assuming that performance varied directly with salary?

4. Set up a performance schedule for this program.
5. Develop a cost accounting system which would equitably charge the departments using the payroll system for the programming, based on standard. Assuming that the programming took 30 weeks for each person, develop a variance.

Chapter XI

METHODS AND PERFORMANCE STANDARDS FOR PUNCHED CARD INSTALLATIONS

INTRODUCTION

Punched card installations, unlike most computer installations, generally exhibit the control functions required for effective management. One reason for this is the slow growth in use of punched cards with us since the Census of 1890. A more important reason lies in punched card technology: the control of punched card machines is largely external, and punched card processing resembles manual clerical processing, in the manner in which the work is organized and controlled.

It is none the less necessary to consider installation of methods and performance standards in punched card installation. Also, most computer installations have some punched card equipment peripheral to the computer.

This chapter will therefore briefly outline the methods standards required in a punched card installation. It will cover only those standards which differ from a computer system, since the latter have already been fully discussed. Performance standards will be covered for operating personnel and for wiring technicians. In a punched card installation, only personnel performance is generally considered important, since equipment performance is almost completely controlled by operator performance.

METHODS STANDARDS FOR PUNCHED CARD INSTALLATIONS

The development of a punched card installation is quite similar to the development of a computer installation. The major phases are:

- Systems analysis and design
- Programming (more often called "wiring")
- Operation

For each of these phases, methods standards should be developed along lines similar to those of preceding chapters. Some of the standards discussed previously are directly applicable; in these instances a reference is made.

Methods Standards: Systems Analysis

Punched card systems analysis is quite similar to computer analysis work. The description of the existing system will generally refer to a manual system, the report layouts will previously have been manual reports and ledgers, and the new systems design will require standards for the following functions:

- Card layout
- Printer layout
- Flowcharting

The job description manual need not be as complex as that produced for a computer system. Program specifications will not be needed; they will be self evident from the card and printer layouts. File retention is usually explained in the card layouts. Card layout standards and printer layout standards are discussed in Chapter III and will not be repeated here. Although flowcharting standards are also discussed in Chapter III, a separate section is included here because of differences in the process.

Flowcharting Standards.—Suggested methods standards are listed below:

1. A complete flowchart must be drawn for each application designed, on 8½ x 11" white bond paper.
2. The symbology to be used in flowcharts is that shown below. [See Figure 11-1.]
3. Each symbol on the flowchart must represent a single operation. The following must be indicated for each operation:
   Card type or electrotype number(s)
   Operation name
   Card columns involved
   Approximate volumes (cards and/or lines produced)
   Panel number to be used (if any)
4. Each symbol on the flowchart must be given a unique step number, consisting of the application letter and a sequential two-digit number, to be assigned in the sequence in which the operations are to be processed.

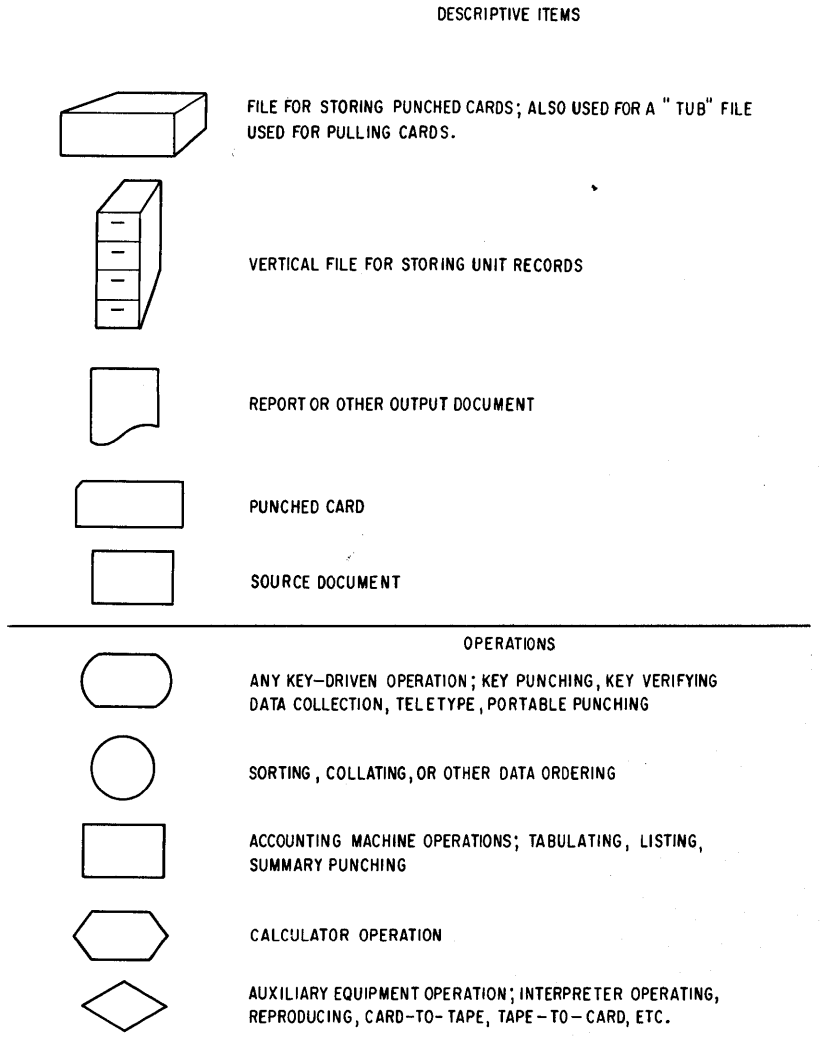


Fig. 11-1. Flowcharting Symbols.

5. Each flowchart shall be accompanied by a separate "procedure" chart, which further defines the operation to be performed. This chart shall include at least one line for each operation step number, and shall include

File to be processed
Origin of the file
Kind of processing
Equipment and card columns to be used
Report produced, and the ultimate disposition of the information

6. All files shall be given a unique file number, consisting of the application letter, a two digit sequential file number, and a one character file description letter, as follows:

M Master File
D Detail File
S Source File
T Table File
H Header Card File
P Pulling Deck

7. A procedure manual will be constructed from the flow and procedure charts, with the procedure chart always opposite its flowchart.

[Figure 11-2 is a typical punched card operation flowchart and Figure 11-3 a form that can be used as a procedure chart.]

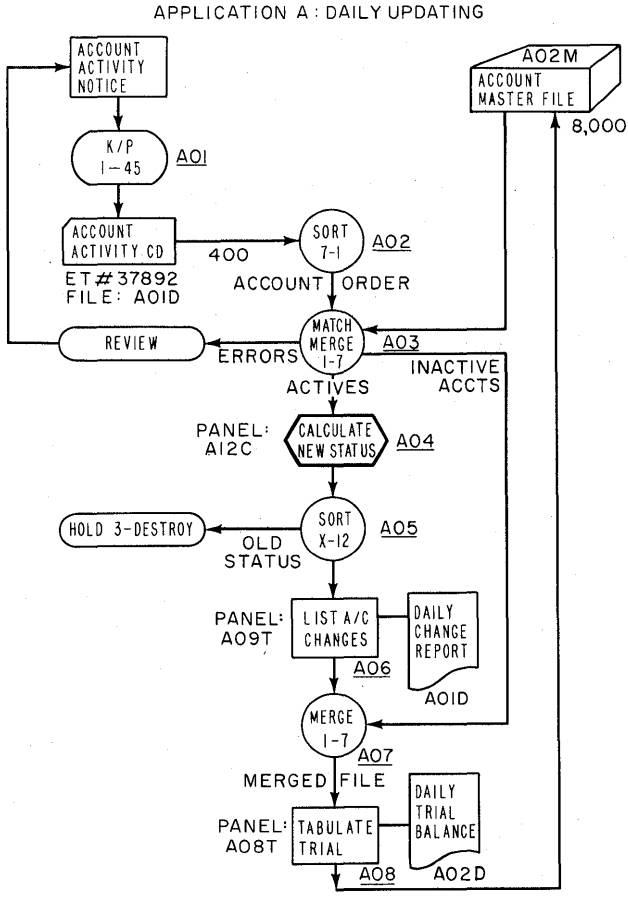


Fig. 11-2. Typical Flowchart.

CLIENT NAME _____
PROCEDURE FOR _____
FREQUENCY _____ PAGE ____ OF ____

STEP NUMBER	EQUIPMENT USED	NAME OF FILE TO BE PROCESSED	ORIGIN	PROCESS	CARD COLUMNS FOR SORTS MERGES	FILE OR REPORT PRODUCED	DISPOSITION OF FILE

Fig. 11-3. Procedure Chart. (*Courtesy*, The Diebold Group, Inc.)

8. Permanent panels shall be assigned a number as follows: the first letter is the application letter, the next two digits a sequential panel number, and the last letter the type of machine for which the panel is used, for example:
   T Tabulator
   C Calculator
9. All reports shall be assigned a report number consisting of the application letter, a two-digit sequential report number, and the frequency code, for example:
   D Daily
   W Weekly
10. Other documentation which must be supplied in addition to the flow and procedure charts is:
   All card layouts and electro-type drawings
   All printer layouts and spacing charts required
   A list of permanent panels required
   A list of reports created

Wiring Standards

The wiring is generally performed by wiring technicians, or programmers. All permanent panels shown on the flow chart must be wired and completely tested. Since each panel performs only one function, such as printing or calculation, the complexity of the wiring task is less than that of its counterpart in computer programming. Wiring standards fall into two categories: standards for guiding the technician in performing the function, and standards for documentation required for the completed panel.

Functional Wiring Standards.—Suggested rules follow:

1. All panels to be wired for a permanent function shall be done with permanent wires.
2. Loose leads must be taped and properly covered.
3. Dominoes must be taped and covered.
4. A proper wire length must always be used. If multiple splits are desired, multiple split wires must be used rather than dominoes.

5. The wiring trays must be kept neat, with each wire in the appropriate bin, as indicated by the sample taped to the edge.

6. Whenever a permanent panel is complete it should be covered and the cover fastened with at least three screws.

7. Wiring tools shall be used to insure that self-contacting panel wires are making the proper contact; the tools must also be used when removing permanent wires—pulling, stripping or plier manipulation is not allowed.

8. All wiring shall be done in accordance with the machine timing charts. A properly wired panel should never blow a fuse on the machine during testing.

9. Filters, selenium rectifiers, or external selectors should not be used on any panel without the specific permission of the supervisor.

10. A complete set of test data must be prepared. One of the first tests to be performed on tabulating panels is done with a set of line finder cards prepared as follows:

Card 1	columns 1 – 9	0000000000
	columns 10 – 19	1111111111
	columns 20 – 29	2222222222
	columns 30 – 39	3333333333
	columns 40 – 49	4444444444
	columns 50 – 59	5555555555
	columns 60 – 69	6666666666
	columns 70 – 79	7777777777
	column 80	8
Card 2	column 1	1
	column 2	2
	column 3	3
	column 4	4
	column 5	5
	column 6	6
	column 7	7
	column 8	8
	column 9	9
	column 10	0
	column 11	1
	column 12	2
	•	
	•	
	•	
	•	
	column 79	9
	column 80	0

The line finder cards will print (in two lines) in the wired report field the column number which is the source of the data.

Standards for Panel Documentation.

1. All panels must be fully documented.

2. Panel documentation shall be kept in a specially designed envelope. [See Figure 11-4.]

CONTENTS CHECK	MACHINE NO._____	APPLICATION_____	PANEL NO._____

	REVISION NUMBER	DATE	BY	CHANGE	DOCUM. CHANGED
TEST DATA CARDS					
LINE FINDER LIST	0				
TEST DATA LIST					
STEP LIST					
CARRIAGE TAPE					
CARRIAGE TAPE LAYOUT					
REPORT LAYOUT					
CARD LAYOUT					

	ALTERATION SWITCH NUMBER								SET UP CHANGE EFFECTED	APPROVALS:
	1	2	3	4	5	6	7	8		
PILOT SELECTORS										
CO-SELECTORS										WIRING
COUNTERS										
LINE SELECTORS										SYSTEMS
PRINT ENTRIES										
COMPARING UNITS										OPERATION
COINCIDENCE UNITS										
AND CIRCUITS										SUPERVISOR
OR CIRCUITS										TEAR DOWN REVIEW
ALPHA STORAGE										
NUMERIC STORAGE										
SWITCHES										

Fig. 11-4. Panel Documentation Envelope Cover.

3. Wiring diagrams need not be made a part of panel documentation, except in the case of extremely simple panels. A more effective form of panel documentation is to list the use made of the various machine components.

4. A usage list shall be prepared for all panels. A separate list shall be made for each of the following components:

Pilot selectors
Co-selectors
Counters
Line selectors
Separate print entries
Comparing units
Coincidence units
"AND" circuits
"OR" circuits
Alphabetic storage registers
Numeric storage registers
Alteration or toggle switches

5. For machines which operate in a serial, stepped manner, such as a calculator, a separate step list will be made up, showing the steps executed in sequence.

6. The panel documentation envelope shall contain the following for each panel:

The usage lists, prepared as indicated in rule 4
The step list defined in rule 5
A deck of test data cards, to be used if changes are to be made
An output listing or deck of the last test, using the test data
A report layout, and a print-out of the line finder deck
A card layout
An extra carriage tape
A carriage tape layout

A form that can be used as a multiple purpose usage list is illustrated as Figure 11-5. A separate step list is shown as Figure 11-6.

Additional standards may be developed for panel documentation, depending on the types of equipment used.

The basic concepts of documentation, and change administration previously described should be retained in developing punched card standards. A revision page may be included with the application systems manual, with the panel documentation, or as a part of the envelope which contains the documentation. The supervisor should file all the materials in one place, under control, so that unauthorized people will not have access to this material. Because of the nature of the material, and because it is infrequently required, retention of duplicate copies may not be warranted.

Standards for Punched Card Operation

The operation of a punched card installation is unlike that of a computer installation. There are several obvious reasons for this, among which are the following:

- The emphasis on one machine in a computer installation is not present; there are many machines, and there are often more than one of the same machine
- Processing requires a considerable amount of card handling; neatness is therefore a much larger problem
- Operators must constantly feed and watch the machines
- Job scheduling is performed by job and operator, rather than machine availability.

Separate operating standards must therefore be made available to the

MACHINE NO._____ APPLICATION _____ PANEL NO. _____

CIRCLE THE APPROPRIATE LIST:	UNIT NUMBER	CONDITIONS (FILL IN TYPE, ETC.)		USED FOR / BY
PILOT SELECTORS				
CO-SELECTORS				
COUNTERS				
LINE SELECTORS				
PRINT ENTRIES				
COMPARING UNITS				
COINCIDENCE UNITS				
AND CIRCUITS				
OR CIRCUITS				
ALPHA STORAGE				
NUMERIC STORAGE				
ALTERATION SWITCHES				
REGISTERS				
FILTERS				
OTHER:				
DRAWN UP BY:				
DATE:				
APPROVED:				
DATE:				
REVISION NUMBER				
0				

Fig. 11-5. Usage List—Variable Conditions.

machine operators in cases where they are responsible for both a computer and punched card equipment.

Housekeeping.—Housekeeping rules do not differ significantly from those for computer installations; the only important difference is that smoking is generally allowed in punched card installations while frequently not allowed in a computer installation. In the latter case, the dust and ashes tend to interfere with the proper operation of the tapes and other electromechanical equipment; in the former case, the interference is less noticeable.

Other housekeeping rules will be the same as the rules developed in

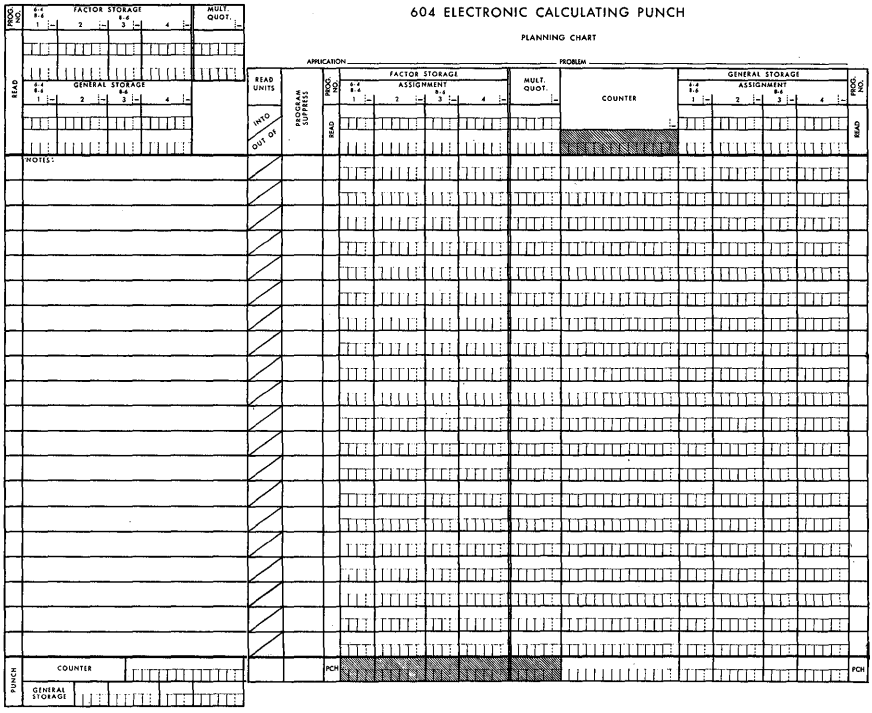


Fig. 11-6. Step List.

Chapter VI. Orderliness, room layout, and the procedures established for the replacement of cards and forms are vital.

Machine Time Logging.—Because of overtime provisions required by the manufacturer, it is extremely important that exact machine logs be retained by machine. This can be done in several ways:

- The use of a manual time record for each machine, as indicated in Chapter VI
- The use of a bar chart recorder with more than one machine attached, as indicated in Chapter VI
- The use of elapsed time recorders.

The last method is illustrated in several forms in Figure 11-7. Included are an elapsed time recorder which merely counts time, a recorder which counts the number of cards processed or the number of lines printed, and recorders capable of recording both time and volume. The latter item is of great value in exact performance evaluation based on volume. They are available for all types of punched card equipment, including key

driven equipment, where the elapsed time is calculated as a function of the *continuity* of punching using a DC relay. This equipment is surprisingly inexpensive, and well worth the installation.

Control Functions.—The control functions used in a punched card installation are not much different than those of a computer room. Report distribution control, data control and data coordination are equally if not more important. Scheduling is often handled differently; this is the only exception.

Scheduling.—Whereas in a computer installation the scheduling is for a single machine or at most two or three machines, in a punched card installation there are many machines, some of which are exact duplicates and may therefore be regarded as the same. Since the performance of each project involves the use of various machines in a pre-determined sequence, and since in each case an operator is required to attend the machine, scheduling must first be accomplished by job and operator; machine loading is generally secondary to the availability of operators and the priority of the projects to be done.

To provide a basic schedule, each job is analyzed to determine

- Volume of operation
- Operations required
- Date the input information is available
- Date the output information is required
- Frequency

On this basis, and on the basis of the established standard times, a

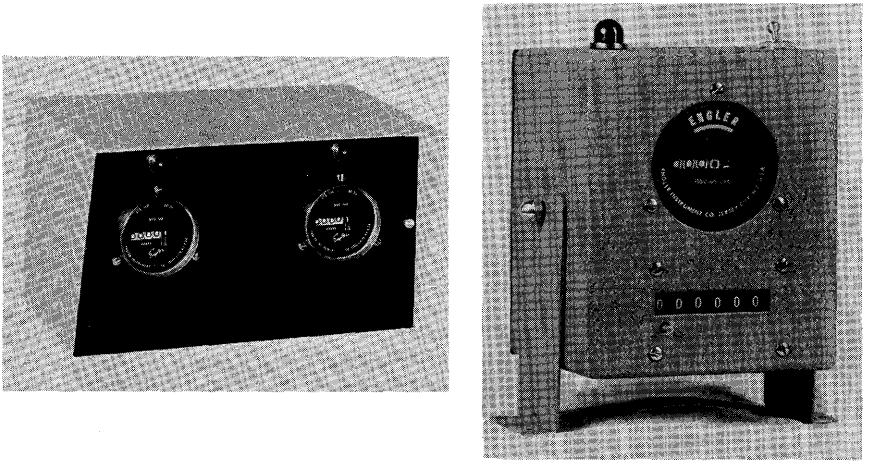


Fig. 11-7. Elapsed Time and Card Counting Recorders. (*Courtesy*, Engler Instrument Company)

project fact sheet is derived. This fact sheet will indicate the times required, the tasks required and the latest day or time that the job must be started in order to complete the final tasks in time. A sample fact sheet is illustrated in Figure 11-8.

The project fact sheets derived for each job are summarized to obtain the total time capacity of each kind of equipment for each day of the month, and for each month. Summaries by day and by month are shown in Figures 11-9 and 11-10 respectively.

The daily summary of projects can be used to formulate a personnel and equipment schedule. A personnel assignment sheet can be prepared, such as is illustrated in Figure 11-11. If preferred, a schedule by machine type and operator can be made up. The choice is largely dependent on whether the installation uses functional operators or project operators. In the former case, each operator operates a specific machine, or series of machines. The schedule will then be by machine type, and will show the sequence of projects which the operator must do on his specialized equipment. This type of schedule is illustrated as Figure 11-12. If the installation assigns operators to follow specific projects completely through, the schedule will merely show each operator to which projects he is being assigned for the period. The operator is familiar with the project, since he is always responsible for it, or he will use the flowchart to supply him with the sequence of steps that he must perform.

USER DEPARTMENT _____	APPLICATION _____			PROJECT NO. _____
	FILE NO.___	FILE NO.___	FILE NO.___	APPROVALS
AVERAGE VOLUME				STANDARDS
MAXIMUM VOLUME				
MINIMUM VOLUME				
SOURCE OF DATA FILE				SUPERVISOR

DATE/DAY INPUT AVAILABLE _____ FREQUENCY _____
DATE/DAY OUTPUT REQUIRED _____ LONGEST TASK _____
LATEST STARTING DATE _____ TOTAL MANHOURS _____
RELATED PROJECTS _____

STEP NUMBER FROM FLOWCHART	TASK TO BE PERFORMED	NO. OF PASSES	MACHINE TYPE	STANDARD UNIT TIME	STANDARD TOTAL TIME
OVER ▱	TOTAL TIMES:				

Fig. 11-8. Project Fact Sheet.

DAY:	DATE:	SPECIAL TIMING:											
PROJECTS REQUIRED TODAY:													
PROJECTS TO BE STARTED TODAY:													
PROJECTS IN PROGRESS:										APPROVAL:			
PROJECTS	PRIORITY	KEY PUNCH 026	KEY VERIFY 056	SORT 083	INTERPRET 557	COLLATE 088	REPRODUCE 519	TABULATE 402	TABULATE 407	CALCULATE 604	OTHER	DECOLLATE	BURST
TOTAL REQUIRED HOURS													
TOTAL AVAILABLE HOURS (8 HOURS / MACHINE)		40.0	16.0	24.0	16.0	24.0	16.0	8.0	16.0	8.0		8.0	8.0
NET VARIANCE													

Fig. 11-9. Daily Project Summary

The latter type of operation generally works better in installations with completely trained operators, familiar with all equipment. It requires less communication between operators, and a great deal less data coordination. If proficiency can only be obtained in a limited number of machines, however, the functional methods works adequately if the number of projects is not too large.

General Machine Operation.—A number of basic machine operating rules can be formulated. Some of these are generally applicable to all machine types; others are strictly a function of the individual machine. Typical of general rules are the following:

1. When starting a job on a machine, remove the panel in place and return it to the proper panel rack, located by number.
2. When completing a job, do not remove the panel. This will protect the panel contact points from dust and other interference.
3. Always reset a tabulator to the top of the page.
4. When finished with a machine, turn power off if no one else is waiting to use it.
5. Always replace excess paper on the stock room shelf.

APPROVALS:		MONTHLY EQUIPMENT UTILIZATION SUMMARY								MONTH: 19			
DAY NO.	DATE AND DAY OF WEEK	KEY PUNCH 026	KEY VERIFY 056	SORT 083	INTERPRET 557	COLLATE 088	REPRODUCE 519	TABULATE 402	TABULATE 407	CALCULATE 604	OTHER	DECOLLATE	BURST
1													
2													
3													
4													
5													
6													
7													
8													
9													
10													
11													
12													
13													
14													
15													
16													
17													
18													
19													
20													
21													
22													
23													
24													
	TOTAL FOR MONTH												
	AVAILABLE IN MONTH												
	NET VARIANCE												

Fig. 11-10. Equipment Utilization—Monthly Summary

6. Always remove carriage tapes from the machine and replace them on the special carriage tape mounting board.

7. Immediately notify the supervisor if errors occur because of machine failure. The engineering log will be filled out completely by the supervisor, so that recurring problems can be rapidly corrected.

8. If the machine has a wired machine stop, always notify the supervisor when the stop occurs.

9. Always perform basic checks on initial data, to insure that the set-up has been made correctly. These checks consist of the following, for the machines indicated:

EAM DAILY ASSIGNMENT SHEET

DATE _____ SHIFT _____

OPERATOR	JOBS TO BE WORKED IN L TO R PRIORITY	REMARKS	JOBS READY

Fig. 11-11. EAM Daily Assignment Sheet. (*Courtesy*, Lockheed-Georgia Company, A Division of Lockheed Aircraft Corporation)

DATE _____ DAY OF WEEK _____ MACHINE NO. _____

SHIFT _____ TOTAL SCHEDULED HOURS _____ MACHINE TYPE _____

SPECIAL INSTRUCTIONS: _____ OPERATOR:

SCHEDULED TIME		DATA COMING FROM MACHINE	PROJECT NUMBER	PANEL NUMBER	DATA GOING TO MACHINE	CHECK COMPLETION	ACTUAL TIME	
FROM	TO						FROM	TO

Fig. 11-12. Functional Assignment Sheet

Machine No. or Type	*Checks To Be Performed*
Card punch	If programmed, punch and review sample cards
Card verifier	If programmed, verify several cards; force errors
Sorter	Review column setting; review selector setting; check to insure that desired counters are operative
Interpreter	Check line setting; pass two cards through and review the interpretation; use line finders if unfamiliar with panel
Collator	When merging, check several merged cards from all pockets; when matching, review the cards in each pocket; force sequence error to insure proper panel
Reproducer	When reproducing, check at least three cards; when gang punching check first to last of a group
Tabulator	Check line-up; review set-up change switches; pass several cards through; clear final total; check hopper stop, last card read out, and all other functions
Calculator	Check the first few cards to insure accurate calculation

Other rules which exemplify good operating practices can be developed in each installation. These rules should be carefully enforced, and should be provided in written form to all trainees.

Emergency, Supply, and Exception Procedures.—All of the above are the same as those outlined for a computer installation, pages 171-174, Chapter VI.

Panel Storage and Control.—To avoid operator time waste in looking for the right panels, or in wiring temporary panels when permanent panels are available, panel storage should be carefully established. Panel racks are almost always required, and are available in all sizes. Each panel should be assigned its own location in these racks, and all panels should be carefully replaced in their proper location, as indicated in the rules below:

10. All panels must be replaced in the special racks provided in the correct size rack and the location established by number. Standard panels (80:80, 90:90, 60:60) must also be stored in a special location; they are numbered starting with a U for utility panel. There must always be available at least one blank panel of each type.

11. All panels shall be tightly covered. No operator may remove a cover or modify a panel. All panel changes shall be made under the immediate supervision of the operations supervisor.

12. Whenever a temporary panel is called for on the procedure chart, the operator shall review the requirement with the Senior Operator or the Assistant Supervisor. If a similar panel is available, or a slight change can be made to an existing panel to fulfill the temporary requirement, the Assistant Supervisor must approve this action. In all other cases, the Senior Operator shall prepare the temporary panel.

13. In the event of machine failure, attributed to panel difficulty by the maintenance engineer, the supervisor shall assign a Senior Operator to review the wiring with the maintenance engineer.

14. Whenever a cover is removed, it must be replaced at the end of the shift, even if the change is incomplete. All changes to panels must be duly recorded in the documentation envelope.

Documentation Control.

15. A central file shall be maintained in the data processing room, containing the following:

Systems manuals containing flowcharts, procedure charts, and layouts for each project or application

Panel envelopes containing the complete panel documentation

Machine manuals showing the operation and wiring of the equipment

16. The central file shall be kept in the following order:

Systems manuals, by application code

Panel documentation, by number

Machine manuals, by machine number

17. A separate supply of cards is to be available, close to the central file, for signing-out any documents.

18. Anyone removing a folder from the central file must observe the following rules:

The entire folder or envelope must be removed; separate sections may not be removed from the envelope.

A person removing a folder or envelope from the file shall substitute in its place an "out" card, indicating on it the date, panel number, machine number, or systems letter, and his name.

19. Changes made to the documentation must correspond exactly to the changes made to the system or the panel; the change and revision number must be approved by the Operations Supervisor.

PERFORMANCE STANDARDS

The approach used for establishing performance standards is generally similar to the approach used in developing personnel standards for computer operation. The overall cycle is the same:

- Development of Standards
- Development of a Schedule
- Measurement
- Evaluation
- Action

There are several areas for which performance standards can be established: equipment utilization, operator performance, and the performance of wiring technicians.

Equipment Utilization

Since equipment performance is largely dependent on operator performance, equipment standards should be restricted to the calculation of a utilization percentage. It is difficult to determine the effectiveness of utilization, since the equipment will usually operate at the rated speeds if the operator performs properly.

The project fact sheet created for each project provides a "standard time" for each step in the operation. This is obtained, as described later, by adding equipment set-up time to the time necessary for the equipment to process the indicated volume, modified by an "operator handling" factor. This value is then posted to a daily and monthly recapitulation, which shows the total theoretical usage of each equipment item.

The effective utilization rate is then determined by dividing the standard time for the assigned work by the available hours. This gives the utilization percentage, based upon the established standard, for a given period as follows:

$$\text{Utilization, } \% = \frac{\text{Total standard time for all work assigned in period}}{\text{8 (hours per day)} \times \text{No. of days in period} \times \text{No. of machines}}$$

This percentage should be graphically recorded on a daily basis, to indicate the utilization factor of each equipment class. An increase in usage for a class of machines may warrant action to prevent future bottlenecks or overtime. This might be the addition of another unit, the replacement of this unit with a faster model, or changeover to a different type of unit.

Similarly, a decrease in utilization percentage might forecast reducing rentals by removal or by reduction in speed, or it might indicate that the machine operators prefer an alternate type of processing. For example, when two types of sorters, collators, or tabulators become available, the utilization of the most popular unit will continually increase and the other type will suffer from operator obsolescence.

Operator and Equipment Performance

Standards for operator and equipment performance cannot be easily separated. Operator performance is a function of how well he keeps the machine supplied; i.e., how close to the rated speed he can make the system perform. The machine generally operates at the rated speed when it is properly supplied with information, otherwise it does not operate at all. For each machine, therefore, a handling allowance is established, for emptying stackers, filling hoppers, and the like. In addition, the

standard includes a small amount of time for set up, as required for the particular machine.

For each of the following machine classes, the standards indicated are suggested as starting points. Total time is expressed in minutes, and key driven operations are expressed in hours, because of their normal length.

Machine Class: Key Driven; Machine Number: IBM 024, 026, 031; Remington Alphabetic Key Punch

Standards for Punching (Including Set-Up):

	Key Strokes per Hour
All alphabetic punching	5,000
Mixed alphabetic/numeric	4,000
Large percentage numeric, little alpha	8,000
All numeric punching	10,000

To calculate the number of key strokes required:

Number of cards $\times$ Average number of columns per card $=$ Total key strokes

(If less than 15 columns are punched per card, add a factor of 10% for card release.)

Machine Class: Key Verifier; Machine Number: IBM 056; Remington Photo-Electric

Time Standards: Same as for key punching but add a factor of 3% for error detection stops.

It is good practice to use different operators when key-verifying cards since this prevents the repetition of errors by the same person. If the number of errors made by the key punch operator is excessive, the time standard for the verifier operator will be unfair, since many more stops will be encountered. In addition, the time standard for the key punch operator must be disregarded, because of the large percentage of errors. Measurement should only be considered if the error percentage is reasonably constant; an increase in error percentage should cancel measurement of both key punch and key verifier and result in some action with respect to the punch operator.

One solution to this problem is to establish a group standard. With this concept, a total productivity standard is established for the key punching and key verifying group, or a productivity standard is set for punching and verifying of the specific project. If, for example, a project consists of 2,000 cards, average 40 columns, all numeric punching, the standard would be set as follows:

$$\text{Key punching Time} = \frac{2{,}000 \times 40}{10{,}000} = 8.0 \text{ hours}$$

$$\text{Key Verifying Time} = 8.0 \times 1.03 = 8.24 \text{ hours}$$

The group standard for the project will be 16.24 hours, which will be used to evaluate the total performance of the "team." If all of the work in the key punch verifier section is lumped into a total group standard, this would provide an evaluation only of the performance of the section as a whole.

The major disadvantage of group standards is that it prevents an accurate analysis of individual employees. This can be overcome to some extent by applying the group standard to individual teams of punch and verifier operators. By determining the performance of each team, and then by varying the team make up, it is possible to arrive at a determination of which operators make the largest contribution.

Machine Class: Sorting;

Machines: IBM 080-1, 080-2, 082, 083, 084, Remington Sorter

Time Standard in Minutes:

$$\frac{\text{Number of cards} \times (\text{Number of columns} + \text{No. of alpha columns})}{\text{Sorter Speed (Modified by handling factor)}}$$

Models	*Sorter Speed*	*Handling Factor*	*Net Speed*
080-1	250 cpm	20%	210
080-2	450 cpm	25%	360
082	650 cpm	25%	520
Remington	800 cpm	30%	620
083	1000 cpm	30%	770
084	2000 cpm	20%	1660

These handling allowances may be modified if the jobs do not allow overlapped stacker removal, or operators are operating more than one machine.

Machine Class: Collating;

Machine Types: IBM 077, 085, 088; Remington Reproducing Collator

$$\text{Merging Time} = \frac{\text{Total input cards}}{\text{Output card speed}} \times 1.20$$

$$\text{Matching Time} = \frac{\text{Total input cards}}{\text{Output card speed}} \times 1.25 + 1 \text{ minute set up}$$

Machine Class: Reproducer;

Machine Number: IBM 514, 519; Remington Reproducer Collator

$$\text{Single Feed Usage Time} = \frac{\text{Number of cards to be gang punched}}{\text{Gang punching speed}} \times 1.05$$

$$\text{Double Feed Usage Time} = \frac{\text{Number of cards to be reproduced}}{\text{Reproducing speed}} \times 1.10$$

In each case a 1 minute set up time should be added.

Machine Class: Interpreter;

Machine Type: IBM 552, 557; Remington Interpreter

$$\text{Time} = \frac{\text{Number of cards to be interpreted}}{\text{Interpreter speed}} \times 1.05$$

Machine Class: Calculator;

Machine Number: IBM 602A, 604, 605, 607, 609; Remington Univac 40, 60, 120

$$\text{Time} = \frac{\text{Number of cards to be calculated}}{\text{Calculating speed}} \times 1.10 + 1.5 \text{ minutes set up}$$

The calculating speed is determined by dividing the maximum speed by the number of cycles each calculation requires; from this it is possible to determine the net card speed in cards per minute.

Machine Class: Tabulator;

Machine Number: Remington Model 3, Univac 1004, IBM 402, 403, 405, 407, 408, 409, 419

Set-up time 2 minutes per form

Operating Time:

$$\text{For Listing} = \frac{\text{Number of lines to be printed}}{\text{Listing speed}} \times 1.10$$

$$\text{For Tabulating} = \frac{\text{Number of cards to be read}}{\text{Tabulating speed}} \times 1.10$$

The operating speed of a tabulator is a function of both the number of cards and the number of lines to be printed. If, for example, the rated maximum speed of the tabulator is 150 cards and 150 lines per minute, a listing job requiring 2 cards per line will operate at a maximum of 150 cards and 75 lines per minute. If the machine is tabulating, it will operate at a maximum of 150 cards per minute. Most machines require only one cycle to perform all of the required functions; included in these functions are reading, adding to counters (tabulating), and printing of one line. If only these functions are performed, the speed of the machine will be the rated speed. If, however, "total" cycles or spacing cycles are required extra time will be used, which will reduce the total rated speed of operation. The simplest way to determine the exact speed of any job is to make an empiric analysis of operation by a simple time study, or by simply measuring the interval to process a known number of cards, or print a known number of lines.

The Univac 1004 operates in an almost asynchronous mode, similar to the stepped functioning of the calculator. As a result, timing must be analyzed more carefully. The rated speed of the card reader, for example, is 300 cpm; the speed may be increased up to 400 cpm by merely releasing the read mode after the first 40 columns have been brought in to the machine. By designing the cards so that all required information is in the front, it therefore may be possible to increase the speed beyond the rated speed.

In almost all cases of calculator and tabulator timing, because of differences between panels, machines, and types of desired operation, it is wise to do a short time analysis. Gross inaccuracies in timing can be rectified: if the job takes far too long, it can be corrected through modification of the panels or through change of the procedures.

Standards for Wiring Technicians

The wiring of a panel is much the same as the coding of a program. It requires a certain amount of advance planning, or logical analysis, to lay out the manner in which the machine's components will be used to fulfill the requirements. After completion of the planning, the actual wiring is done, much like the coding of a program, thus fulfilling the analysis and translating it into the language of the machine. Completion of the wiring leads to testing, sometimes preceded by the preparation of the requisite test data, although live data is usually available.

After the testing has been completed, the panel must be documented, much as a program must be documented.

The tasks that must be performed, therefore, are:

- Logical analysis
- Wiring
- Testing
- Documentation

Wiring is generally somewhat simpler than programming. The size of the task is usually smaller, since punched card equipment is basically unifunctional, and does not integrate functions as a computer does. The analysis is usually expressed in a simpler format and takes hours rather than days. Testing is performed on-line: the panel is mounted and tested and corrections are made on the spot.

Wiring Parameters.—The parameters that affect the time required to wire a panel are similar to the parameters defined for programming. The first parameter, complexity, is completely unchanged; viz.,

A Simple Panel
B Medium Panel
C Difficult Panel
D Extremely Difficult
E Most Complex

The size parameter cannot be expressed in the same terms. The most suitable unit of size is the number of wires to be used in the panel. These may be counted by counting the open hubs, subtracting from the total hubs available and dividing the difference by two, or by giving the technician a standard complement of counted wires, and counting all of the wires left over. A third method is to weigh the panel before and after wiring; the difference is the weight of the wires, which can be translated into an approximate count.

For easier handling, the size unit is expressed as the number of wires divided by 100 and rounded. This parameter is estimated in advance and is validated by a count or weighing at the completion of the panel.

Other parameters could be included separately, or as part of the complexity parameter:

- Number of card formats used
- Number of report formats produced
- Number of steps to be used in a serial step machine

For the sake of simplicity, the only parameters considered in this evaluation will be size, complexity, and a machine component factor which reflects the inherent difficulty of the specific panel and the number of components available. Component factors for representative punched card systems are as follows:

	Machine System	*Factor*
Interpret:	IBM 552	1
	IBM 557	2
Collate:	IBM 077	2
	IBM 085	3
	IBM 088	4
Reproduce:	IBM 514	3
	IBM 519	5
Punch:	IBM 521	7
	IBM 541	7
Read/Punch	IBM 533	9
Sort:	IBM 101	8
	IBM 108	10
Tabulate:	IBM 402, 403	10
	IBM 407	15
	IBM 408, 409	16
	Univac 1004	16
Calculate:	IBM 602A	12
	IBM 604, 607, 609	14
	Univac 40/60	15
	Univac 120	16

This factor assesses the inherent complexity of the panel to be wired, while the other two factors, size and complexity, assess the difficulty of the specific project.

Relationship between Tasks and Time.—The following formulations have been derived to establish the relationship between the tasks outlined above, the above parameters, and the time required to complete the job.

Task 1—Analysis and Assessment

The first task involves:

Familiarization with the problem
Review of the requirements
Analysis of layouts
Development of preliminary component usage

The time required to accomplish this task is expressed as follows:

	Hours for Level of Complexity	
	0	A
	½	B
¼ hour for each machine factor unit *plus*	1	C
	1½	D
	2½	E

Task 2—Wiring

Included in this task are the functions of:

Wiring the panel
Noting in preliminary form the component usage

The time required is a function of number of wires, and of the task complexity:

	Man-Hours for Level of Complexity	
	0	A
	1	B
1 hour for each unit of size *plus*	2	C
	3	D
	4	E

Task 3—Testing

Panel testing involves

Preparation of required test data
Punching of a line finder card set
Testing of the panel
Correction of the panel

Time is a function of size, complexity, and a slight factor to reflect the machine component complexity:

½ hour for each unit of size *plus*

¼ hour for each machine factor *plus*

Man-Hours for Level of Complexity	
0	A
½	B
1½	C
2½	D
4	E

Task 4—Documentation

Panel documentation includes

Envelope
Usage charts
Carriage tape, and its layout
Test output listings

	Man-Hours for Level of Complexity
	½ A
	1 B
¼ hour per unit of size *plus*	1½ C
	2 D
	2½ E

Summary:

Assume that an IBM 407 panel is required, with an estimated complexity of C, and an estimated wire count of 850:

Task 1	15 × ¼ + 1	=	4.75 hours
Task 2	8 + 2	=	10.00 hours
Task 3	4 + 15 × ¼ + 1½	=	9.25 hours
Task 4	2 + 1½	=	3.50 hours
	Total time		27.5 hours, or 3.5 days

Assume that an 088 panel is required, complexity A, with approximately 150 wires required:

Task 1	4 × ¼ + 0	=	1 hour
Task 2	1	=	1 hour
Task 3	1½	=	1.5 hour
Task 4	¼ + ½	=	.75 hours
	Total time		4.25 hours

It should be noted that the machines with a lower machine factor generally also have a smaller panel, therefore accommodating fewer wires. As a result, the wiring time is low; testing and analysis time are relatively much higher.

Development of a Schedule

For the operating department, schedule development has already been discussed under methods standards for punched card operation. For the wiring technicians the schedule developed can be quite similar to the "bar chart" schedule set up for programmers in Chapters IX and X.

Gathering of Data

There is no difference in the methods used to gather data in a punched card operation and a computer operation. The machine log, a separate operator progress report or operator schedule, may be used as illustrated

earlier in this chapter. The wiring technicians should use a reporting technique identical to the reports turned in by programmers, broken down by panel number and task.

Evaluation

Evaluation of wiring technicians corresponds exactly to the method of progress reporting and individual evaluation outlined in Chapter IX. Evaluation of operators is somewhat different, since the data on which their evaluation is based is in a different format. The basic concept is the same. Actual performance is compared to the standard to come up with an efficiency factor, broken down by

- Project, to evaluate project performance
- Operator, to evaluate personnel
- The entire punched card installation

Measures of Quality

Each installation must establish measures of quality to accompany the quantitative measures established. Each supervisor must establish requirements for minimum acceptable quality, or he must develop quality standards which modify the quantity performance standards. The following are suggested as possible measures of quality for a punched card installation:

Key punching:	Number of errors detected by key verification should be less than 1 in 1750 key strokes
Key Verifying:	There should be no errors remaining after verification
Sorting:	The number of sequence errors detected in a sorted file should be 1 or less for each 20,000 cards
Reproducing:	There should be no uncorrected comparing errors
Interpreting:	Information should be 100% correct
Tabulating:	There should be no "reset check" or "auto stop" indications that remain uncorrected; forms alignment should be within 1/32nd of an inch; there should be no card jams
Calculating:	There should be no detected errors
Collating:	Subsequent sequence checks should indicate no further errors.

Other measures of quality can be introduced for the overall quality of output, such as the total number of final reports rerun, the number

of balance proof errors detected at the end, or the number of punching errors found on the final output.

CHAPTER SUMMARY

Using the precepts developed in preceding chapters, this chapter outlines methods and performance standards for punched card installations. In many instances the standards are similar; in others the methods used vary and completely separate standards are desirable, even in cases where a punched card installation and a computer installation operate side by side.

Methods standards for punched card installations include the systems analysis function, the wiring function and operation. Performance standards are developed for equipment and operator performance, and guidelines are drawn for the development of performance standards for wiring personnel. The chapter organization parallels the organization of the book, in that it represents a condensed view of the standards developed for computer installations. This condensation is possible because the scope of punched card operation is generally smaller than the scope of computer operation.

Questions for Review

1. What are the significant differences between punched cards and computers?
2. Why does this contribute to the differences in standards? What are other contributing factors? Why do you feel that punched card standards are necessary?
3. Develop a documentation control system for a punched card installation which shared its documentation with a computer group.
4. Flowchart a payroll system through a punched card installation. Develop the necessary rough specifications, and rate the size and complexity of each of the panels required for all of the machines.
5. Develop the total time required to wire and document these panels.
6. Set up a schedule to accomplish the wiring of these panels.

Appendix A:

JOB DESCRIPTIONS IN DATA PROCESSING

Manager of Electronic Data Processing

Reports To: Comptroller or other senior officer
Supervises: Supervisors of all operating shifts
Programming Supervisor
Systems Analyst

Responsible for the analysis, feasibility study, installation, scheduling and administration of the company's electronic data processing operation.

Determines the equipment specifications and requirements; determines the personnel requirements and prepare the budget and expense reports on activities subject to his authority.

As a member of the Electronic Committee, he participates in policy making decisions affecting the mechanization of applications within the company. If it is the company's policy to sell a part of the availability of the equipment it is his function to act as representative of the company and to maintain customer satisfaction through proper scheduling.

Programming Supervisor

Reports To: Manager of Electronic Data Processing
Supervises: Programmers
Coding Clerks

Is responsible for the design and maintenance of the framework in which the data processing system operates. Must be thoroughly familiar with the applications for which the sytsem is used and the methods employed.

Under his supervision are programmers and coding clerks who carry

out the creation of the system as designed by him. Must be familiar with all forms and inputs in use and with all applications under consideration.

Is responsible for the selection of personnel to fill jobs in the programming area, and for the establishment of *operational* and developmental standards under which the system will operate.

Systems Analyst

Reports To: Manager of Electronic Data Processing

Responsible for the examination of existing and proposed systems, in order to evaluate the relative economies. Responsible for the overall design of the system in conjunction with the Programming Supervisor. Documents and outlines the system preparatory to programming.

Since the incumbent must work independently of schedules and pressures, he should not report to the Programming Supervisor.

Programmer

Reports To: Programming Supervisor
Supervises: Coding Clerks

Must be thoroughly familiar with the complete problems to which the computer is applied; must be capable of preparing a methods analysis or feasibility study. Must act independently under control of the Programming Supervisor, subject to the standards previously established.

Must be capable of drawing flowcharts and block diagrams, outlining both the current and the proposed systems. Must take into account the scheduling, volume of activity and the elapsed time required to perform the operation in order to establish comparative costs.

Is responsible for a part of the supervision of the coding clerks. Must arrange for and supervise the testing of all proposed operating systems.

Coding Clerk

Reports To: Programming Supervisor

Given technical assistance and instructions by Programmer.

Is responsible to commit to memory the language in which the computer operates and must be familiar with the library of subroutines available from the manufacturer and the installation. Must be familiar with the capacities of the computer system in order to translate into machine language the instructions received from the programmer.

Must be temperamentally suited and have aptitude and interest for performing detailed and complicated clerical work.

Is not required to understand the overall logic of the complete system nor the economics of automation.

Operating Supervisor (Console Supervisor)
Reports To: Manager of Electronic Data Processing
Supervises: Console Operators
Tape Librarian

Responsible for the effective utilization of equipment and personnel and for the conformance to schedules and error correction procedures as previously established. Responsible for the maintenance of standards of accuracy and quality of output in accordance with the requirements set down. Must be familiar with all forms, reports and inputs used.

Is required to act independently on situations of an operating nature which are not anticipated. His authority includes the direction of all operating personnel, including the maintenance of logs of operation, and supervision and control of the tape library.

In a multiple shift operation he is required to turn over the operation to the next shift supervisor in a manner enabling transition without interruption.

Console Operator
Reports To: Operating Supervisor

Under direction of the Operating Supervisor the incumbent carries out operations according to operating manuals created by the Programming Group. Must be thoroughly familiar with tape file operation, tape splicing and handling techniques, computer console operation, peripheral equipment operation.

Is responsible for the maintenance of the operating log and should be able to recognize malfunction and determine whether the malfunction is caused by a machine failure, a program failure or a data failure.

Tape Librarian
Reports To: Operating Supervisor

Is responsible for the maintenance, external labeling, orderly filing and assembling of tapes for each job that must be handled by the installation. Is also responsible for the maintenance of error free available ("scratch") tapes, with appropriate beginning and end markers.

Is responsible for the current maintenance of all program files and their coding; of all operating manuals and their coding; of all special control cards used in connection with each job; of all program lists used by the operating group.

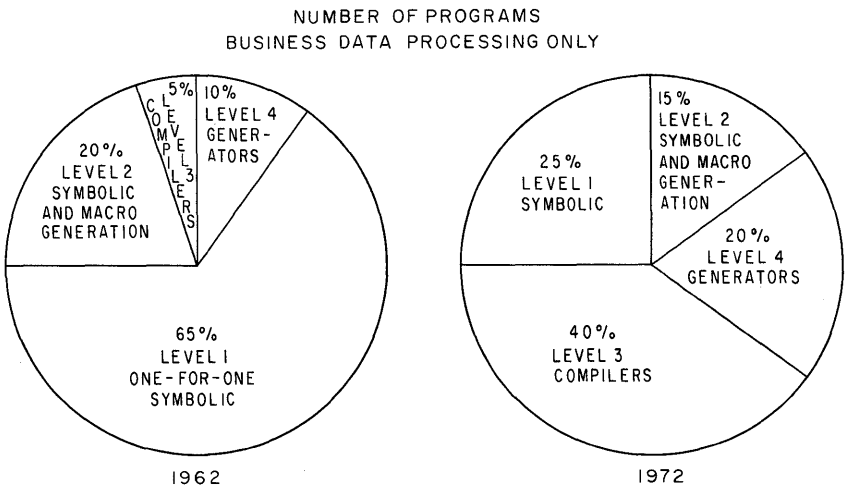
The incumbent must be constantly aware of the contents of the tapes, and must use them in such a way as to prevent inadvertent destruction or alteration of the information recorded.

Appendix B:

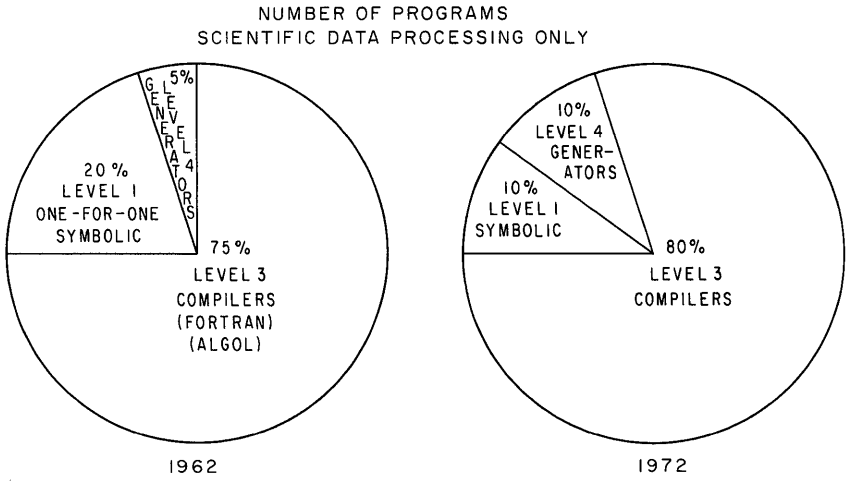
LEVELS OF PROGRAMMING LANGUAGES

		LOWEST LANGUAGE LEVEL SYMBOLIC ONE-FOR-ONE	2ND LANGUAGE LEVEL SYMBOLIC & MACRO GENERATION	3RD LANGUAGE LEVEL COMPILER (COBOL) (ALGOL)	4TH LANGUAGE LEVEL OBJECT GENERATORS (INTERPRETIVE & PROGRAM)
GOOD PROGRAMMER	OBJECT PROGRAM EFFICIENCY	95%	90%	60-85%	45-65%
	PROGRAMMING COST (INDEX)	100	90	40-60	20-40
"BAD" PROGRAMMER	OBJECT PROGRAM EFFICIENCY	70%	75%	50-70%	40-60%
	PROGRAMMING COST (INDEX)	110	100	50-70	30-50

I. Relationship of Language Level to Cost and Efficiency



II. Language Level Utilization Estimates



III. Language Level Utilization Estimates

Appendix C:

PERSONNEL REQUIREMENTS

TABLE C-1
PREVIOUS WORK EXPERIENCE
(20 Corporations)

Previous Job Classification	Percentage Distribution of EDP Employees			
	Total	*Administrative & Supervisory*	*Planning & Programming*	*Operation*
Accounting and Professional	35.5	44.9	43.5	9.6
Administrative and Supervisory	13.3	40.6	11.9	9.8
Tabulating and Keypunch	13.1	2.9	4.4	40.6
Posting, Checking and Filing	10.7	2.9	9.3	17.5
Statistical	5.4	—	6.0	4.7
Correspondence and Secretarial	2.0	—	1.7	3.7
Non-clerical	1.2	1.4	1.1	1.4
New Employees	18.9	7.2	22.1	12.7
Total	100	100	100	100
Numeric Distribution	915	69	637	209

Source: Bureau of Labor Statistics, Bulletin No. 1276.

TABLE C-2
PERSONAL TRAITS REQUIRED
(16 Corporations)

Trait	Number of Corporations Listing Trait	
	Programmers	*Operators*
Trait		
Report writing ability	15	6
Rapid arithmetic	11	4
Observation	12	9
Understanding of underlying principles	16	13
Rapid coordination	0	12
Rapid manipulation of small objects	0	9
Adaptability to a variety of duties	14	11
Responsibility for planning	15	4
Adaptability for repetitive operations	5	13
Judgment based on quantity	13	6
Judgment based on quality	12	8
Adaptability to deadlines, pressure	12	14
Adaptability to precise standards	13	15
Knowledge of theoretical mathematics	4	1
Knowledge of calculus	5	1
Knowledge of accounting	11	8
Knowledge of decimals, fractions	11	8

Source: Bureau of Labor Statistics.

Appendix D:

SAMPLE STANDARDS MANUAL

Manual of Data Processing Standards

for

Financial Publishing Company

Prepared by

The Diebold Group, Inc.

and

Financial Publishing Company

Revision Page

Date	By	Section	Page	Revision

Table of Contents

Chapter I

General Information

General Information

Data Processing Standards

Standards are rules set up and established by authority to provide meaningful information in a consistent, usable form. When complex and detailed problems are analyzed and solved in a similar manner, standards are necessary for the useful exchange of information. Standards are used to give us a guide for gathering pertinent information which is needed. From the initial statement to production operation, standards establish the procedures to investigate, develop, and solve a problem. Standards are called for when problem solving techniques or groups of problems are related by a common need. To those who must analyze and solve problems or operate complex machines, standards are the source for information and instruction to carry out this work.

In data processing and the use of computers, standards are necessary for the successful computer shop. Because of the many detailed steps to be taken before a problem is ready for production operation the need for gathering and recording pertinent information accurately and in sufficient detail, becomes important. A programmer in making an analysis gets into ramifications of a problem which, in the days before computers, were the proper business of several people. No one considered it necessary for a single person to become acquainted with all the ins and outs of a job because the manual or mechanical operations formed their own division of responsibility. The computer unifies all these aspects, thus the need to know all the details which must be recorded in a useful manner.

In setting Data Processing Standards a method of operation is developed which sets guide lines for obtaining the necessary information. This method is the sequence of steps followed in gathering information and the techniques developed for using a computer to solve a problem. Standards relieve the programmer of the many little bookkeeping decisions by establishing a uniform way of assigning program labels and organizing the program.

This manual of Data Processing Standards, the foundation for exchange of information among all people in data processing, provides:

(1) a common frame of reference,
(2) effective communication among people,
(3) effective communication between people and machine,
(4) clear cut definitions of terms,
(5) personnel responsibilities, and
(6) procedures of operation.

The manual is the programmer's working tool. The programmer is asked to learn and understand the use of these standards. The successful programmer masters these precepts.

Scope of this Manual of Data Processing Standards

The manual, designed to explain the details of programming and operating standards is divided into five major sections.

Program Analysis and Problem Definition

This section is an outline of steps taken to solve a problem offering a description of development from the original problem definition to programming, and a tabulation of information to be recorded.

Programming Standards

The standards and conventions of programming are defined here. The most important reference is to flow charts and labels. Each programming task is explained and related to logical program development.

Documentation Standards

Documentation is the orderly collection and arrangement of information about a program. The Program Manual gives the interested person information about every aspect of the problem and reflects the care, understanding and ability of the programmer.

Operation Standards

Standard practice for machine use and the function of the program library are described.

Appendix

The appendix contains programming techniques, and glossaries of computer terms, Financial Publishing Company terms and standard abbreviations for documentation.

Manual: Care and Use

Additions and revisions to this manual will be made only with approval of the Company Secretary. Suggestions may be directed to the Secretary for review. Each programmer will be responsible for keeping this copy current as additions are made.

You are expected to understand and apply the standards established in this manual. Your success as a programmer is measured by the success of your program, by the understanding, exposition and organization of the information in your program manual, and by the simplicity and clarity of your operating instructions.

General Information

Equipment Specifications

Specifications of the IBM 1401 Data Processing System at Financial Publishing Company:

1401 Processing Unit, Model B-3
1402 Card Read-Punch
1403 Printer, Model 2

Special Features

Core Storage, 4000 Positions
Multiply-Divide
Index Registers
Store A Address
Store B Address
Move Record
High-Low-Equal Compare
Additional Print Control, 132 print positions
Numeric Print Control
Interchangeable Chain Cartridge Adapter
Early Card Read

Supporting Equipment

Key Punch	026
Card Verifier	056
Sorter	082
Reproducing Punch	514

Software Programs

SPS - The Symbolic Programming System
Prelist
Assembly
Postlist
Memory Printout
Card-to-Card Utility (80-column reproducer)
Card-to-Printer Utility (80-column list)
D.CAL System
D.CAL 1-Input Data Validation Program
D.CAL 2-Card-to-Card Calculation Program
D.CAL 3-Card-to-Card Calculation and List Printer Program
D.PCH - Memory Punch Routine
FARGO
RPG
JOCAL

Chapter II

Problem and Program Development

Problem and Program Development

A problem evolves into a program through a series of 6 steps:

(1) Problem Definition
(2) Numerical Analysis
(3) Program Definition
(4) Program Identification
(5) Input and Output Design
(6) Program Development

The first five steps consist of gathering necessary facts and displaying them in reasonable order for use in step six, programming. The logical order of these steps should be followed in gathering facts, developing in approach, and cataloguing the necessary information in a manner and form which permits both a check list and quick reference for needed information.

Problem Definition

A written description of the problem defines the major considerations and the special characteristics affecting the problem solving approaching. Information included in this step will include:

(1) Purpose
(2) Input
    Source, form, sample
(3) Output
    Form, sample
(4) Special Characteristics
(5) Problem Glossary
    Defines terms unique in meaning to this problem

This general statement of the problem describes elements to be developed in detail by the program definition.

Numerical Analysis

The numerical analysis of arithmetic operations show the factors, where they come from (input from card or computed by program), the arithmetic operations, and the intermediate and final results. When a formula is used it should be stated in the most general form, in the specific form to be used, and the transformation from one to the other. Equations using a representativ example of the calculations show the actual arithmetic operations. Indicate the ranges and limits of the factors imposed first by the problem and second by the capacity allowed for in the program. Show where adjustments in the

answer are made; for example, the rounding 5 to adjust half cents or greater up to the next cent. This is a fact gathering section; the sequence and manner to be used by the program is determined in the Program Definition.

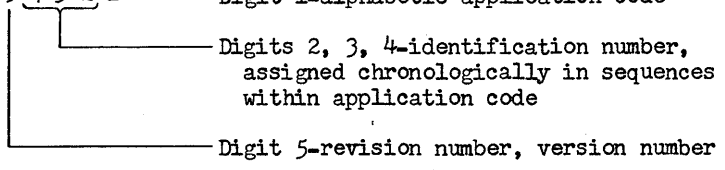
Program Definition

The development of program specifications from the problem definition produces a program definition. Several programs developed from a single problem definition form a program system. The program definition is explicit in the detail information required, the mathematical techniques, the control, the inputs and outputs and its relationship to other similar programs. Step three will cover the following information:

(1) Purpose
(2) Input
    describe fields on card layout forms; type; source; composition; limits; volume
(3) Output
    describe field on card layout and print layout forms; type, source; composition; limits; volume
(4) Factors
    examples of the calculations; specifically the sequence and manner the program will use in calculations
(5) Special considerations
    restrictions; exceptions; effect on current practice; relationship to other programs
(6) Program glossary
    defines terms unique in meaning to this program

Program Identification

Identification numbers will be assigned to all programs. The assignment will be made when the exact definition has been established. The identification is a five-digit alphanumeric number in the following format:



The SPS card allows five columns, 76-80, for the identification number to be punched. This is done in the source deck and is automatically carried

through to the object and condensed decks. If the source cards for one program are used in the development of a second program care should be exercised to get the old program number out and the new one in.

The alphabetic application code used as the first character is:

A Accounting
B Bond values
C Consumer finance
P Insurance premium charts
S Schedules
T Mathematical, financial tables

Example: A revision to program B0015 will be numbered B 0016; if B0016 includes B 0010 through B 0015, its number will revert to B 0010.

Input-Output Layout

Layouts will be prepared for the formats of all inputs used and outputs produced by the program. This shall be done, without exception, before programming. The detail of these layouts may be changed during the design of program logic and reanalysis of program specifications. Neat, accurate documents reflecting the current format of inputs and outputs must be available; if a change is made they provide the point of departure. These layouts will be incorporated in the program documentation.

Card Layout

On the IBM Card Layout form used to illustrate all card input and output fields, describe fields by name, field limits by vertical lines. In the space on the form give the following information, in as much detail as necessary:

Column	Size	Field Name	Description	Source	Data Type	Consistency Limits
					Alpha	limits of field
					Numeric	limits of codes
						limits of range

Describe briefly the card preparation procedures for each kind of card.

Printer Layout

On the 1403 Program Planning and Testing Format for High Speed Printers, illustrate all printed output, identify by the report name and the program number that produces it. Show the following on your printer output layout:

(1) Write in preprinted headings, underline
(2) Write in emitted headings
(3) Write two sample lines for each type of detail printing
(4) Show punctuation for every edited field
(5) Indicate maximum value in each field
(6) Show all total levels
(7) Show entire vertical and horizontal dimension of form
(8) Emit headings and identification for output on blank paper

Program Development

The steps of problem analysis described in the preceding pages provide the necessary information for developing a program. Programming is composed of the five tasks: logic design, coding, test preparation, program testing, documentation, considered and executed sequentially. The importance, detail and complexity of these tasks require standards defined in Chapter 3, Programming Standards.

Logic Design

Three stages of program logic must be completely designed and documented before any subsequent programming tasks are begun.

1. Analysis and program organization-analysis of the program definition to establish specifications for designing the computer oriented logic.

2. Design of the general flow of program logic-MACRO flow chart-the major logical elements, illustrated in a flow chart, required to meet program specifications.

3. Design of the specific logical processes-MICRO flow chart-the individual logical steps, illustrated in a flow chart, within each major logical element. Micro flow charts show the connecting linkage.

A third level of flow charts, called the ABSOLUTE flow chart, represents each instruction by a separate symbol to explain and illustrate a complicated logical structure.

A summary of the three flow chart levels makes the following distinctions:

Macro Flow Chart - one symbol for each major logical function; one symbol per routine.
Micro Flow Chart - one symbol for a series of instruction within a routine; one symbol for several instructions.
Absolute Flow Chart - one symbol for each instruction.

Coding

Coding is the statement, in machine-acceptable language, of the logic described in the flow chart. Coding must be written legibly and accurately to minimize key punch errors. You will follow this sequence in which basic program components will be organized to permit the standardization of program continuity:

Program Description
Standard Halts and Non-Standard Halts
Housekeeping Routines
Input-Output Routines
Main Logic
Work Areas
Special Subroutines

Test Preparation

Test data, used to verify the accuracy and reliability of a program in meeting the requirements of program definition, shall be developed during the first two programming tasks. As decisions, logic tests and control functions are considered, test data shall be prepared to verify logic and coding. Test data should be reviewed by someone familiar with the problem definition to insure that no obvious condition has been overlooked. Test cases shall be used for desk checking the program logic. The flow of logic will be checked by processing simple test data in the dry run mode. Key punch, coding and logic errors can be detected in this way. You are responsible for preparing the test data and organizing the test plan.

Program Testing

A program with test data and operating instructions is submitted to the operations section for test (or assemble or list as the case may be). Sufficient past test information should be called for to provide a clue to your trouble if the test failed. This includes memory dump, register addresses, and the output. You should, after each test, record the results and measure your progress with regard to the overall test plan. When all requirements are met, the test data and test output are retained in the program documentation. To be sure you have understood and satisfied all the program specifications, review the test results with the program director.

Daily test sessions are provided for testing and assembling programs. This period, under the supervision of the machine room, is not a time for programmers to operate the 1401. Programmers may make arrangements with the Operations Supervisor for machine time.

Documentation

Descriptive material is compiled throughout problem analysis and program developments. The final task is to review, organize and edit for accuracy and legibility all this information included in the permanent program record. Important points of logic should be clearly defined; the purpose and capabilities of the program described in non-technical language. All this documentation is collected in a Program Manual, and includes:

Problem Definition
Program Definition
Flow Charts
Input, Output Layouts
Formulas and Equations
Assembled Listings
Test Data and Test Results
Operating Instruction
Program Abstract

Chapter III

Programming Standards

Programming Standards

Programming standards, established in the common program areas of flow charts, organization, coding, testing, provide consistent, reliable programming techniques. These standards are the starting point for a good program.

Flow chart terminology and practice, adapted for computer use, acquire new meaning and application. Our symbol definitions when used intelligently gives the necessary information demanded of a flow chart.

Mnemonic or uniquely systematized labels are not built on the general, underlying principles required to accommodate a large number of varying labels. Standard labels eliminate the need for each programmer to build his own system. Private label systems are incomplete, quickly become meaningless, change between programs, and defy the exchange of information. Private labels will not be used.

Each section in Chapter 3 is important; you should understand and practice them all in each problem you do. They eliminate the unrelated detail associated with the clerical tasks of programming,- free the programmer for the important job of problem solving. Your success will depend on how you understand and accept these principles.

Logic Design-Flow Charts

A flow chart, the picture representation of a problem to be put on a computer, and the basic flow of information shows the sequence of operations performed by a computer in the execution of a problem. An operation usually consists of a number of program steps.

A flow chart must have a set of symbols to show the following:

(1) The kind of operations to be performed.
(2) Changes in the sequence of operation as the result of a test.
(3) Changes in the sequence of operation as the result of modifying an instruction.
(4) Explanatory information and initial conditions.
(5) Connection links to a part of the problem which is continued on another page.

Two kinds of flow charts are referred to in this manual. The macro flow chart shows: the major operation symbols in their logical context within the general construction of the problem; the major decision points, inputs and outputs, major starts and halts; the over-all considerations of the problem and how they tie in and relate to each other. This is the initial statement of what the problem is and how to solve it.

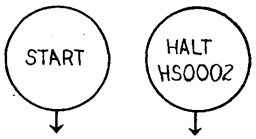
The micro flow chart develops in detail the contents of each major operation symbol. A major operation involves the logical operations of arithmetic storing, comparing, branching. The micro flow chart shows with symbols and explanation the precise logic to solve the problem. Each symbol and its contents must give the necessary information so that the program may be written directly from the micro flow chart. A well prepared micro will be so clearly drawn that a programmer may write the program without hesitation. The work of a program is in the flow charts; the coding is a matter of writing only.

A third level, referred to as an absolute flow chart, shows each program step in a symbol. To explain a complex decision operation this degree of detail may be necessary.

A flow chart identification system assigns an alphanumeric code to each symbol. This code becomes part of the branch label to insure a direct correlation between coding labels and flow chart symbols. The micro flow charts are developed at a level allowing the programmer to code directly from it even though each symbol represents more than one instruction.

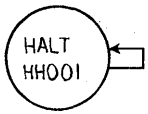
Flow Chart Symbols

The following flow chart symbols shall be used.

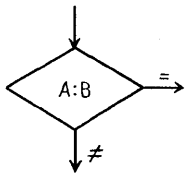


Start Stop Symbol

This symbol is used to indicate starting and stopping points in the program

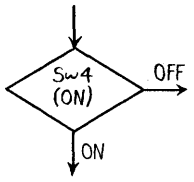


For a "hard" halt (no further processing), the arrow returns to the circle.

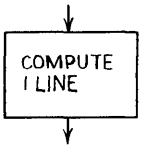


Test Symbol

This symbol is used for all test and branch instructions. It indicates a logical choice in the program based upon a decision. The normal flow of logic should be from the top to bottom, the exception going to the right.

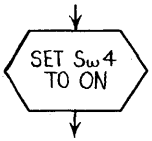


A special use of this symbol is to test the condition of a switch. The initial state of the switch is shown in ( ) inside the symbol.



"Do" Box

This is the all-purpose symbol to illustrate specific operations.



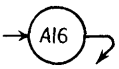
Set Symbol

This symbol indicates setting of switches, index registers, and address modification. It does not include within it the limit test which normally follows or precedes this symbol.

Input-Output Symbol

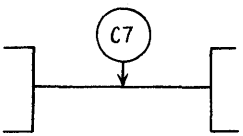
This symbol indicates all input-output operations: read, print, punch.

Connectors

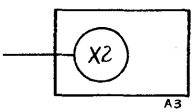


This is a connector; the arrow points generally in the direction of the entry. The next operation is A16.

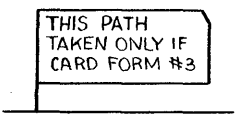
The next operation is on page 3, symbol C7.



This is an entry, normally placed out of the main logic flow.

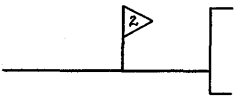


This symbol indicates transfer to a programmed routine which includes the return transfer back to the original point of exit in the flow-chart. In this example, X2 is the start of a routine which, when completed, will transfer back to A3. The routine at X2 may be large and complex.



Flags

This is an assertion box used to provide more information than is given in the flowchart symbols.



Index register flag, index register number in the flag. This symbol is used whenever index registers are used.

Symbols help to graphically illustrate a problem and how it is solved. Decision points and input-output operations are easily found, since these are the most likely places for changes. The symbols are not sufficient in themselves, however, and pertinent documentation in the symbols, and flags, must be used.

Flow Chart Conventions

(1) The Standard Symbols shall be used to describe logical functions.

(2) The Standard financial and data processing notations and authorized abbreviations shown in the Appendix shall be used. A legend page must define all exceptions to these standards.

(3) A macro flow chart shall be drawn for every program. It shall identify and illustrate every logical operation with a separate symbol. Each program shall be divided into at least five and no more than 17 logical operations; the macro flow chart shall be diagrammed on one sheet of 8 1/2 × 11 inch white bond paper.

(4) An alphabetic A-Z code shall be assigned in sequence to each symbol on the macro flow chart. The following letters will not be used: I, Ø, Q, U, and V. These are either reserved for special purposes in the Standard Label System or are restricted because of the tendency to confuse them with other handwritten alphabetic or numeric characters.

(5) A micro flow chart shall be drawn for every significant operation shown on the general flow chart. It shall illustrate the logical process that occurs within each major operation. This level of flow chart shall specify the linkage between operations. Each micro chart may contain up to a maximum of 99 symbolic figures and shall be on one side of 8 1/2 × 11 white bond paper.

(6) Each micro flow chart shall be identified by the alphabetic code of the symbol on the general chart which it represents. Every symbol on the micro chart shall be assigned a two-digit numeric code 1-99. Every symbol is identified by a three-digit numeric code.

Digit 1 - Alphabetic code assigned to macro flow chart symbol
Digit 2-3 Numeric code assigned to micro flow chart symbol

(7) Every page shall have an identification block (2" × 3") in the upper right-hand corner of the page. It shall contain

program name and number
block letter and functional name
programmer's name

(8) Both levels of flow charts must accurately and consistently represent the program logic. They must be updated with all changes made.

(9) All connectors and entry points must be shown and identified. All input and output operations shall be shown as distinct symbols on the macro flow chart and shown in micro flow chart form when other than a single instruction is required.

(10) Each symbol shall contain a brief description of the logic it represents. Standard notations and abbreviations shall be used. (see Rule 2)

(11) A closed subroutine shall be illustrated as a separate operation.

(12) Flow charts shall read from top to bottom.

Program Organization

General groups of operations such as housekeeping, input, output, main logic, work areas, accumulators, counters, and constants which occur in every program, shall be found in the same place in program coding sheets.

The SPS card allows two digits for page numbering: The following page numbers are assigned to specific operations.

Coding Sheet Sequence

(1) Page 00 Comments:
- Program title and identification
- Programmer name
- Assembly date and number of assembly
- Console set up conditions
- Components used, punch, reader, printer
- Radial stackers if other than normal
- Punch and normal read
- Special 1403 instruction
  - carriage control tape
  - alpha numeric print chain
  - 6 or 8 inch vertical spacing
- Restart address
- Halts identified by label, I and A address and cause
- Comments cards describing the program at some length
- Input cards identified by type, name, code

(2) Pages 01-10
- Housekeeping
- Standard Halts
- Normalizing of counters, switches, connectors, accumulators
- Print registration routine
- Memory print out routine

(3)	Pages 11-20	Input; cards read, tested for sequence or code and the input stored in work areas
(4)	Pages 21-30	Output; transferred from compute work areas to print areas, edited and printed
(5)	Pages 31-80	Main Logic
(6)	Pages 81-89	Work areas usually set by a DCW. A separate page for each category of work area should be used.
(7)	Page 90	ACCUMULATORS and COUNTERS are work areas set aside in which to do arithmetic operations.
(8)	Pages 91-99	CONSTANTS such as numbers, edit words and alphabetics to be printed in the heading.

Memory Layout

Input-output areas, index registers, and machine timing bits occupy fixed positions of core memory. Other positions are assigned to constantly used program operations. The 4000 positions of core are assigned as follows:

From	To	Use
0000		Reserved for machine read timing and parity check
0001	0080	Card Read Area
0081		Record Mark, set by program
0082	0086	Parameter Card Count
0087	0089	Index Register 1
0092	0094	Index Register 2
0097	0099	Index Register 3
0010		Reserved for punch timing and parity check
0101	0180	Card Punch Area
0182	0186	Line Print Count
0187	0199	Machine Utilization Routine
0200		Cleared by print area clear
0201	0332	Print Area
0333		Blank to clear print area
0334	0337	Standard Halt 1
0338	0341	Standard Halt 2
0342	0345	Standard Halt 3
0346	0349	Standard Halt 4
0350	0353	Standard Halt 5
0354	0357	Non-Standard Halt
0358	0361	Non-Standard Halt
0362	0500	Memory Correction
0501	3998	Housekeeping, main program, constants and work areas
3999		Not available

Coding

Coding involves writing instructions from flow charts; assigning labels to branch points, areas, constants, halts; and writing comments, punched in cards along with the instructions, to describe constants, steps and operations in the program.

Character Writing Standards

Numbers and characters may be misread by key punch operators when penciled coding sheets are being keypunched unless care is used in writing. The following style of writing numbers, alphabetics and special characters will be used.

Numbers

1 2 3 4 5 6 7 8 9 0

Alphabetics

A B C D E F G H I J K L M N O P Ø R S T U V W X Y Z

Special Characters
(These eleven are standard on keypunch)

&	12	ampersand (plus zero)
,	0-3-8	comma
.	12-3-8	period
□	12-4-8	lozenge
-	11	dash (minus zero)
$	11-3-8	dollar sign
*	11-4-8	asterisk
/	0-1	slash
%	0-4-8	per cent sign
#	3-8	pound sign
@	4-8	at sign

In addition to the 10 numbers and 26 alphabetic characters the 1401 understands 28 additional special characters.

The 6 bit binary coded decimal arrangements permits 64 different code patterns. Special characters are names given to different code patterns, apart from the numbers and the alphabet, which are usually odd multipunch codes in the card. If punched into a card they will be read by the machine.

They may be used in arithmetic but will give hash results. They may be compared, moved, or used as a d-modifier; some will print, all will punch, and some are operation codes.

The special characters listed below are in low to high collating sequences:

	Card Code			Card Code	
b		blank	,	0-3-8	comma
.	12-3-8	period	%	0-4-8	percent
⌑	12-4-8	lozenge	=	0-5-8	equal, word spearator
(	12-5-8	left parenthesis	'	0-6-8	apostrophe
<	12-6-8	less than	"	0-7-8	tape segment mark
≢	12-7-8	group mark	¢		cent (program generated)
&	12	ampersand	#		pound sign
$	11-3-8	dollar sign	@		at sign
*	11-4-8	asterisk	:		colon
)	11-5-8	right parenthesis	>		greater than
;	11-6-8	semi colon	✓—		tape mark
-	11	minus	?		plus zero
/	0-1	slash	!		minus zero
△	11-7-8	delta	≠		record mark

The following letters will not be used to tag a flow chart symbol:

I	too easily confused with 1 (one)
Ø	too easily confused with O (zero)
Q	too easily confused with O (zero)
U	too easily confused with V
V	too easily confused with U

Labels

Labels identify program entries in coding. SPS allows for a 6 digit alphanumeric label which becomes a machine address in a subsequent operation. All of our labels will be 6 digits.

The following table, describing our label system, provides maximum information about the referenced area, branch point, constant, or halt; it is directly related to the flow chart identification; it provides for every possible label.

STANDARD LABEL SYSTEM

Character	Char.								
1 Area Z	2	A-Accumulator C-Counter P-Punch card area		R-Read card area T-Print area W-Work area				X-Index Register	
	3-4	Size of area; positions of core							
	5-6	Sequence number							
Branch point B	2	Block letter from macro flow chart							
	3-4	Block number from micro flow chart							
	5-6	Sequence number							
Constant K	2	A Address Constant	E Edit word	H Heading	L Literal	M Message	N Numeric	S Switch	T Table	X Others
	3-4	Size of constant								
	5 6	Sequence number	Number of decimals Sequence number	Heading line No.	Sequence Number					Size, value of constant or sequence number
Halt H	2	H-absolute (hard), or dead end halt; S-intermediate, soft halt								
	3-6	Sequence number								

Character	1	2	3	4	5	6
Area	Z	A	1	2	0	7
Branch point	B	C	1	9	0	3
Constant	K	A	0	3	0	4
Halt	H	H	0	0	0	1

Area

An Area is a section of memory, defined by a DCW, DC, or DS statement, used for storing input-output data, results of calculations, or a field in which to perform arithmetic.

An Area may comprise:

ACCUMULATORS to add, subtract, multiply, or divide.
COUNTERS, a special form of accumulator, used for sequence numbering and loop control. Usually they will be only of two or three digit capacity.
PUNCH CARD AREA, the storage area of a field prior to being transferred to a punch field. The linkage between the punch card area and the punch field would constitute a variable type of output format.
READ CARD AREA is the storage area to which a field is transferred after card read and before processing.
PRINT AREA is the storage area of a field prior to being transferred to a print field. The linkage between the print area and the print field in columns 201-332 constitute a variable type of output format.
WORK AREA, the storage area for holding temporarily results of arithmetic operations.
INDEX REGISTER, a special type of counter.

Branch Point

A BRANCH POINT is the instruction to which the program branches as the result of a successful test.

Character 1 is B
2 is the macro flow chart block letter
3-4 is the micro flow chart block number
5-6 is the sequence number

Constants

A constant is a value or field which is not subject to change. The exception to this is a switch which may be in one of several states depending on action taken as the result of a test.

ADDRESS CONSTANT is a 3 position core storage address defined by a DSA.
EDIT WORD is the control field containing the punctuation for the printed output data and the zero for automatic control of zero suppression. KE 0821 b,bb0.bb
READING is alphabetic and numeric information for printing headings on blank paper prior to tabular listings.
LITERAL is an alphanumeric constant.

MESSAGE is alphabetic and numeric information which is subject to print under program control to inform the operator of a condition such as "last card has been processed", which can be determined by program testing.
NUMERIC is a numeric constant.
SWITCH is a one position field which can be tested and altered as a result of test.
TABLE is a list of values in order by some argument to be referred to for a particular value in the list.
OTHERS which do not fit this scheme, to be identified by an X in character 2 and the size, or value, or name of the constant in characters 3-6. Any program entry which defies the standard label system may be assigned a KX xxxx label.

Halts

Standard Halts shall be located in memory in positions 334-353. All Halts shall be four-position instructions. The A operand of absolute halts (hard halts) shall contain the address of the halt instruction's operation code. The A operand of temporary halts (soft halts) shall contain the address of the instruction to which the branch is taken after corrective action. An absolute halt condition precludes any further processing; a temporary halt condition indicates that a processing stage has been completed and that operator intervention may be necessary before processing can be continued.

There are five Standard Halts:

Memory Positions	Label	Instruction	Condition
334-337	HH 0001	H 0334	Final Halt
338-341	HS 0002	H ----	Restart
342-345	Hb 0003	H ----	Sequence Error
346-349	Hb 0004	H ----	Error
350-353	HS 0005	H ----	Print Registration

Non-standard halts may be developed by the programmer when necessary. They must, however, conform in format and content, and will be assigned memory locations following the standard halts. Non-standard halt labels shall be numbered in sequence with the standard halts. The second character of the halt label shall be H to indicate hard halt; S to indicate soft halt.

Comment Cards

Comment cards are to be used throughout the program. At the beginning they will describe the problem, list options and switches, describe forms, input cards and stacker selection, output punched cards and stacker selection. This will also be part of the run manual. Do not abbreviate in comment cards.

Interspersed in the program, comment cards will describe segments of the program, and what loops go in or out of the segment.

In debugging put in an unconditional branch to the next segment at the end of each segment; these are to be removed before final assembly of object deck. After this unconditional branch DCW descriptive cards of the segment will show in the memory dump as comments. These are to be removed before final assembly.

The switches indicated in the first paragraph are the last card switch and any internal switches set by a card.

Comment cards appear only in the listing, taking no space in memory. The listing will be part of program documentation, the original coding sheets will not.

Comments cards, preceding the coding, will specifically describe the following:

(1) Each routine represented by a macro flow chart symbol.
(2) Special programming and mathematical techniques.
(3) Each phase of a multi-phase program, summarizing the purpose and relationship.
(4) Significant input, output, work areas.
(5) Decision, control and exception processing.
(6) The first page of coding will contain the following information:
Program name and number
Programmer name
Brief description of problem
Brief description of program
Formulas
Special codes
Special instructions
Input and output devices used
Identification of input
Identification of output
Restart address
Halts, cause and summary of corrective action
High-order position of memory used
Console setting
Estimated average run time

Program Testing

Program Testing answers these two questions: Have we met the job requirements? Have we made an accurate translation in programming? Testing is done for (1) clerical errors such as wrong keypunching, incorrect constants, insufficient area and counter capacity; (2) interpretive errors in coding from the flow chart; (3) logic errors in the flow chart; (4) validity errors in the original understanding and statement of the problem.

Clerical errors plague us all and can best be eliminated by careful writing and thorough preliminary planning of all the factors involved in arithmetic, storage, editing and printing.

Interpretive errors evolve from insufficient documentation of micro flow charts. The micro chart should show enough detail, in both words and symbols, to permit coding directly without stopping to work out further logic. Elimination of this kind of error comes with using and writing flow charts. A good flow chart explaining exactly what is to be done is difficult to draw.

Logic errors result from a misunderstanding about how an operation works or how it ties into subsequent operations. It usually comes from trying to imply too much in an operational symbol of the micro or, just the opposite, from an absolute flow chart of every instruction which gives no clear-cut separation of operations. Complete certainty about the operation and accurate diagramming will eliminate logic errors.

Validity errors involve the entire problem logic as distinguished from the preceding errors which are concerned with program logic. Understanding of the entire problem field and the relative position of your program in the field will greatly reduce the chance of this kind of error.

Testing begins once a program has been coded. Desk checking of each segment, SPS pre and post list, and label table are all first level aids in weeding out clerical, label and interpretive errors. Testing will progress along the flow of macro logic. Take advantage of memory print outs, interrogate memory, and stop at the end of a macro logic block to print out positions of core. Testing a program for accuracy and validity can take as much time as all the other tasks.

Testing

Considerable care should be exercised in the preparation of sample data. Test cases should be constructed to allow progressive checking of each logical branch within the program. It is vital that test data include all necessary code punching and control fields. The volume of data should be held to a minimum to facilitate testing. Actual data should not be used until artificial test samples run correctly. Repetition of data should be avoided, as it wastes machine time and increases the time needed for preparation. The block diagram should be checked using cases prepared to test all possible conditions. These cases should then be used for testing.

Although the output of one run may become the input to a subsequent operation, it is unwise to plan on using this output, until the program is fully tested.

Location of program errors is not an exact science. It is a matter of applying logical reasoning processes to the available evidence determining which program steps did not accomplish the expected result. The best approach is to locate a point at which the program was functioning correctly, then check each succeeding step up to the error halt. In many cases, the real

cause of the error will be found at a point in the program much earlier than the stop. The error may have created a condition, such as unintentionally changing an area of program storage, which caused a subsequent operation to be performed incorrectly.

A methodical approach will accomplish far more than random theorizing as to possible causes. Examination of the memory printout will usually give a quick answer as to the incorrect condition causing the halt. The problem is to find out what created the incorrect condition. Tracing the program step-by-step to locate the error is a tedious task and sometimes quite frustrating. In many cases, a person other than the programmer can locate errors faster and easier. The best rule is to assume that every step is wrong until examination of the contents of storage proves otherwise. Reference to the block diagram and listing of input simplifies the job.

Progressive checking should start with the simplest example of input data to prove out the main flow of logic. Once we have arrived at the end with correct figures for the very simplest case we then progressively add, one at a time, each condition planned for in the program. In the micro block this will be a test of both exits of a decision symbol. In the macro block this will show the variations of input and output.

A test plan will be used to focus your attention on the step-by-step sequence of operations to be checked. Often, it is enough to know you arrived at a certain address. Other times you will need to know the results of arithmetic. Work out a schedule based on success but with check points along the way listing the address or results you expect. Use the test plan form. Record your progress. When the program is completely tested, summarize the number of tests, the number of errors, and what they were.

Common Coding Errors

1. Presence or absence of word marks as a result of prior operations
   Absence of B-field word mark in compare operations
   Improper setting and clearing of word marks
2. Edit control word shorter than field to be edited (data field)
3. Loss of high-order zone bits when modifying addresses by subtraction
4. Different length fields when comparing
   Signs in units position causing incorrect compare
   Signed fields compared to unsigned fields
5. Key punch errors: 2 and Z, 5 and S, 1 and I, 0 and ∅
6. Address arithmetic and character adjustment errors due to faulty counting or program revisions
7. Undefined symbolic operands
8. Lack of A-field WM for Move and Suppress operation
9. Improper setting and clearing of record mark

10. Incorrect labels on instructions causing wrong indexing in assembly
11. Constant areas overlapped with program
12. Attempting to partially chain operations other than Move and Load
13. Wrong d-modifier
14. Failure to provide exit from a loop under all conditions
15. Confusion on Hi-low-equal compare
16. Last card test
17. Arithmetic overflow
18. Loop control counter off by one

The following documents will be complete, accurate, and available before any test data is run.

(1) Flow charts
(2) Card, storage and print layouts
(3) Carriage control tape
(4) Operating instruction
(5) Sample output data
(6) Utility routines needed
(7) Test data
(8) Test plan
(9) Source deck for assembly or object deck for test

Testing Procedure

You will follow the following procedures when submitting programs to the machine room for test or assembly.

(1) Fill out a green TEST card and put in front of deck.
(2) Source and assembly decks will be punched in SPS cards. Test data cards will be plainly marked TEST DATA, program # _____.
(3) The control card must be punched with the date and programmer's initials in columns 40-55.
(4) Operating instructions if more detailed than called for on the TEST card.
(5) A memory print out shall be produced at the end of each test run.
(6) Test assembly periods are scheduled once a day. Programmers are not permitted to operate the 1401 during these test periods. Programmers will have access to the machine after all daily work is done. Arrangements must be made with the machine room supervisor.
(7) All test and assembly material will be returned to the programmer by the machine room.
(8) Programmers will review their testing and programming accomplishments at the end of each week.

Chapter IV

Documentation Standards

Documentation Standards

Program documentation presents in accessible and intelligible form all relevant information concerning a computer program.

This information is used for different purposes according to the individual needs of those reading the documentation for instruction. The machine operation will be concerned with operating instruction and input key punching instruction. Company management is interested in an abstract describing the program, its limits, and its capabilities. A programmer wants to find clearly drawn flow charts and documented listings to speed changes and additions. The program director needs this in a neat, clean, presentable package.

All this material must be collected in logical, usable sections to be readily accessible without limiting its effectiveness. This is your responsibility.

Contents of Program Manual

The program manual is divided in three sections: general information, operating instructions, programming. It contains all the documentation. In outline form, the following summarizes the content.

I. General Information

(1) Title page, showing program name, number, programmer, date completed
(2) Revision page
(3) Table of contents
(4) Problem definition, a written description
(5) Problem analysis, an analytic description, showing: input factors; numerical analysis; examples of all calculation, size of all factors involved, rounding, limits of numbers; formulas - a relationship expressed in algebraic symbols; equations - a relationship expressed between factors and results; variables defined.
(6) Program abstract

II. Operating Instructions

(1) Machine set up: console, reader-punch, printer
(2) Carriage control tape
(3) Forms and cards to use
(4) Programmed halts: I, A address, reason for halt, corrective action
(5) Restart address

(6) Operating instructions
(7) Average run time
(8) Sample of output
(9) Input key punch instructions

III. Programming

(1) Program definition
(2) Macro flow chart
(3) Micro flow chart
(4) SPS post list
(5) Label table
(6) Subroutines used
(7) Memory layout
(8) Print spacing chart
(9) Input, output card layout
(10) Test data
(11) Sample content

The outline summarizes the minimum required information included in the program manual but this does not preclude other useful information developed in the course of programming. If switches are used extensively, include a list describing this function. If a multipurpose program, show relationship between factors. The criteria of additional information is that it be useful.

Final review and approval of the finished program manual will be done by the program supervisor.

Operating Manual

The Operating Manual (the first two sections of the Programming Manual) is the best source of information that the computer operator has. If he is to do a good job running the program we must furnish him with pertinent information, and we must anticipate difficulties he is likely to encounter and give him instruction for corrective action.

The success of a program is reflected by the operation. A good job of programming means simple, well directed instructions for preparing the card input, precise machine set up directions, and proper identification of the output. These factors contribute significantly to an accurate product prepared consistently and reliably.

Disposition of Documentation and Program Cards

Two copies of the Program Manual and one copy of the Operating Manual will be submitted to the program library when the program has been completed. One copy of the Program Manual will be kept in the library. The second copy will be kept in a separate location for safe keeping. The Operating Manual

will be kept in the machine room.

Program Cards

Source Deck	:	one interpreted copy of each deck
Assembled Deck	:	will be filed in the programming department
Condensed Deck	:	four copies of the condensed deck will be made, two copies for the machine room, one copy to be filed in the programming department, one copy will be kept in a separate location for safe keeping

All program cards will be labeled: on the first card with program identification number, date, F/C; on the last card with program identification number, L/C; on the top with program identification number. The condensed cards will be color coded:

Accounting	A	blue
Bonds	B	salmon
Consumer Finance	C	green
Premium Charts	P	yellow
Schedules	S	pink

A carriage control tape will be labeled with the program identification number and filed in the machine room.

Program Abstract

A program abstract, summarizing the documentation, gives essential information to non-programming members of the company. It will be brief, to the point, and contain the following information:

Program name and number
Programmer's name and date finished
Purpose and features
Mathematical approach, formulas, equations
Relation to other programs
Input and output

Four copies of the program abstract will be typed; they will go to the following places:

One to each of the two program manuals
One to a file of abstracts in the program library
One to a file of abstracts for general reference

Subroutine Documentation

In addition to information called for in the program manual, subroutine documentation will include:

1. Subroutine Name
2. Purpose
3. Description
4. Programming Procedures

    a. Subroutine call
    b. Entrance requirements
    c. Entrance
    d. Exit conditions
    e. Execution time
    f. Length of coding
    g. Constants
    h. List of other subroutines referred

5. Operating Procedures
6. Flow-charts
7. Coding
8. Program halts

Chapter V

Operating Standards

Operation Standards

The machine room supervisor, guided by these needs, will schedule and control the computer operation:

(1) one day's work be done before another is started
(2) schedules, consumer finance and bonds be run each day
(3) test time be provided

The machine shall be attended while running and even effort made to keep it running continuously with a minimum of lost time.

Programmers submitting new programs to be listed, assembled, tested or run, will use a green TEST card giving full instruction for each request. Production runs will be submitted with a blue PRODUCTION card giving the set up, estimated volume and time, and disposition of finished work.

Computer operators should become familiar with the operating details of each program and have a general understanding of its purpose. The Operating Manual, kept in the machine room, should provide this information and provide corrective steps when an unusual situation develops. Incorrect or inadequate information should be referred to the programming supervisor for correction or clarification.

A machine time record will be kept, by day, showing the kind of work, the amount done and the time to do it. Included in the kind of work are the following: production; test time; down time due to machine malfunction; maintenance; training; time lost due to data error, operator error, program error; ribbon change. Time is measured in hours and minutes.

Program Library

The Program Library will provide program numbers and will be the place where documentation of the program is kept. Material kept in the program library will be:

(1) Program Manuals
(2) Program Cards
(3) Program Abstract-third copy
(4) Program Change Record-second copy
(5) Program Index
(6) Inventory of Jobs in Process
(7) Machine Time Records

Programmers will remove, change, and correct this material only with the permission of the supervisor.

Program Change Record

Any alteration to an approved program will be done only with the program supervisor's permission. Reason for the need and method of change will be reviewed with the supervisor before the change is made. A Program Change Record will be followed and added to the Program Manual. A second copy of the Program Change Record with the detail program additions and deletions will be filed in the Program Library to be added periodically to the second copy of the Program Manual.

Chapter VI

Appendix

Chapter VI

Key Punch Program Card

The 026 Print Card Punch uses a program card to control duplicating, skipping, alphabetic and numeric punching, field definition, and left zero print.

The card code to do these are:

Card Code	Function	Card Code	Function
12	field definition	1	alphabetic shift
11	start automatic skip	2	left zero print
0	start automatic duplication	3	print suppression

For keypunching the SPS cards we have two program cards, one for instruction cards, the second for constants and work areas where the processor assigns the address. The second card provides for a sign punch in column 23.

All identification and page, line fields should be punched. All punched columns should be printed.

SPS INSTRUCTION CARD

Col	Punch	
1	0, 2	(S)
2	12, 2	(B)
3	2	
4-5	12, 2	(B)
6	11	
7	12	
8	1, 22	
9-13	12, 1, 2	
14	1, 2	
15-27	12, 1, 2	
28	1, 2	
29-39	12, 1, 2	
40	1, 2	
41	12, 1, 2	
56	11	
57-75	12	
76	0, 1	
77-80	12, 1	(A)

SPS CONSTANT AND WORK AREA CARD

Col	Punch	
1	0, 2	(S)
2	12, 2	(B)
3	2	
4-5	12, 2	(B)
6	2	
7	12, 2	
8	1, 2	
9-17	12, 1, 2	
18	11	
19-22	12	
23	2	(to force Ampersand to print)
24-39	12, 1, 2	
40	1, 2	
41-55	12, 2	
56	11	
57-75	12	
76	0, 1	
77-80	12, 1	

Chapter VI

PROGRAM ABSTRACT

Program _____ Identification _____

Programmer _____ Date _____

Purpose of the program

Capabilities of the program

Mathematical description; formulae and equations

Reference to other programs

Configuration requirements; core size of program and special features

Type of input and output

Standard subroutines used

Chapter VI

PROGRAM CHANGE RECORD

Program _____ Identification _____

Programmer _____ Effective Date _____

Change initiated by _____ Change Number _____

I. Description of change:

Memory correction Change in logic Supervisor's initial

II. Check list for revision By Date

Machine language working

SPS Source deck

Object deck

Condensed deck

Listing

Micro flow chart

Manuals

Test data

Tested Before and After

Test results approved

Other changes

Previous documentation updated _____

Program decks replaced _____

Librarian _____

Date _____

TESTS: Date _____

Programmer: _____

Program #: _____

Sense switch A on: _____

Final Halt - I address: _____

Restart address: _____

Machine Operator: _____

PRE-LIST: _____

ASSEMBLE: _____ Iterations: _____

TEST: _____

Memory dump- BEFORE: _____ AFTER: _____

Punch output: _____ card type: _____

Tape # _____

Paper _____

Space 6 8

CONDENSE:

IF FINAL HALT IS INCORRECT, RECORD DISPLAY LIGHTS ON REVERSE SIDE

PRODUCTION: Date in: _____

Chain: Alpha _____ Numeric _____

Program # _____

Input Source: _____

Punch Output: _____

Card Type _____

Estimated Run Time: _____

# of Pages _____

Customer _____

To be Photographed: _____

Form: _____

Paper Alignment: Left Slot _____

Paper Edge: _____

Wire Guide: _____

Tape # _____

Return to _____

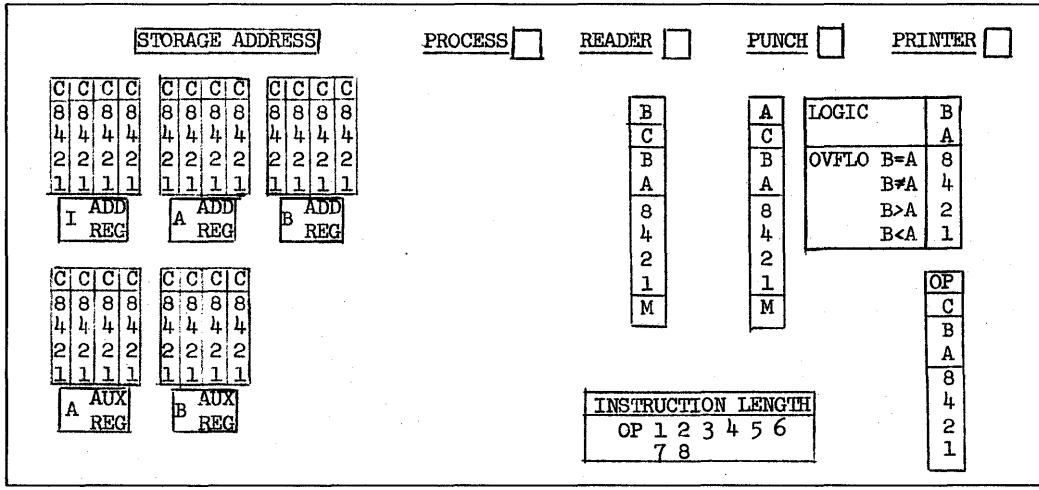
Final Halt - I Address: _____

Restart Address: _____

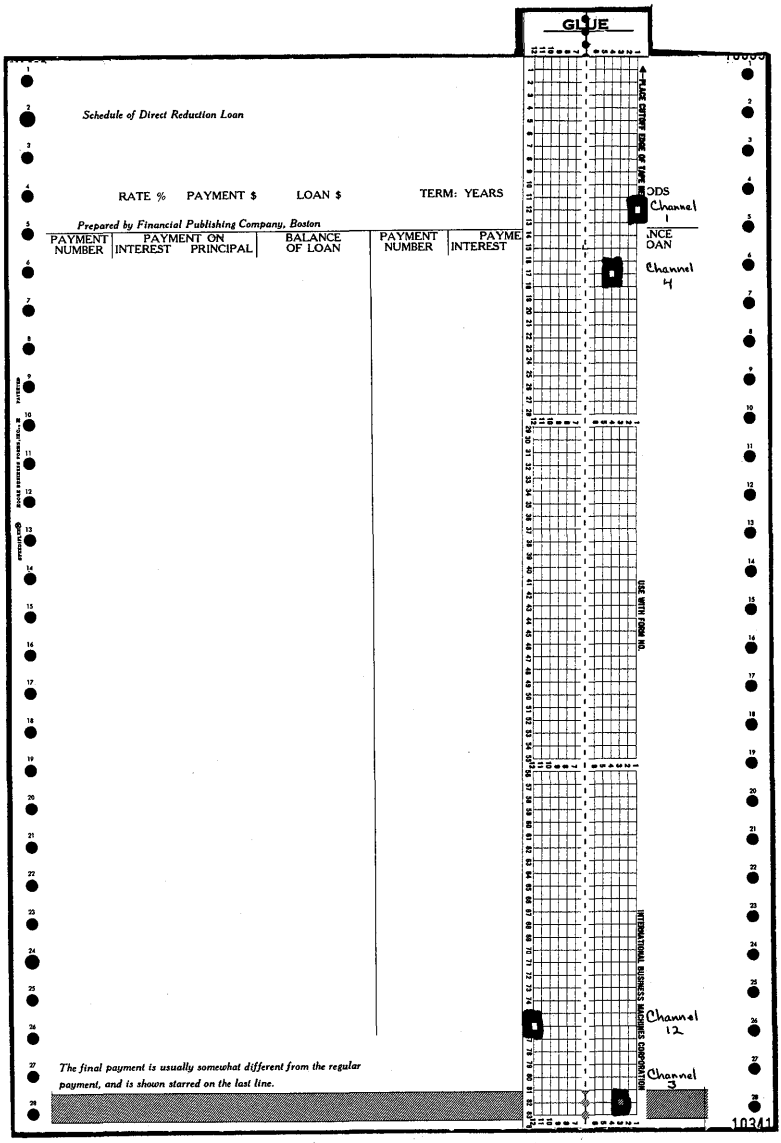
Machine Operator: _____

Date Out: _____

If final halt is incorrect, record display lights on reverse side



Chapter VI


GLUE
Schedule of Direct Reduction Loan
RATE % PAYMENT $ LOAN $ TERM: YEARS
Prepared by Financial Publishing Company, Boston
PAYMENT NUMBER
PAYMENT ON INTEREST
PRINCIPAL
BALANCE OF LOAN
PAYMENT NUMBER
PAYME INTEREST
The final payment is usually somewhat different from the regular payment, and is shown starred on the last line.
Channel 1
Channel 4
Channel 12
Channel 3
10341

Chapter VI

MATHEMATICAL SYMBOLS

+	plus or positive
-	minus or negative
+ -	plus or minus positive or negative
- +	minus or plus negative or positive
x or .	multiplied by
÷	divided by
=	equals
≠	does not equal
≡	congruent
>	greater than
<	less than
⩾	greater than or equal to
⩽	less than or equal to
≃	similar
∴	therefore
$\sqrt{a}$	square root of a
$\sqrt[n]{a}$	n^{th} root of a
a^n	n^{th} power of a
Σ	summation of
f( )	function of
c( )	contents of
Δy	increment of y
|a|	absolute value of a
∪ a	a, presence of a, as in Venn diagram
∩ a	no a, absence of a, not minus a, as in Venn diagram

Chapter VI

MATHEMATICAL SYMBOLS OF FINANCE

%	Percent
$	dollars
¢	cents
@	at
P	Principal present value
j (P)	Nominal rate (P conversion periods per year)
i, j, r	Rate of Interest
S	Compound amount of $1 for a periods $S = (1+i)^n$
n	Number of periods
V^n	Present Value of $1 (n periods); $V^n = \frac{1}{(1+i)^n}$
R	Annual rent

Index

Other Important VAN NOSTRAND Books of Interest

MANAGEMENT STANDARDS FOR DATA PROCESSING
By DICK H. BRANDON, *Brandon Applied Systems, Inc.*

AN INTRODUCTION TO AUTOMATIC COMPUTERS:
A SYSTEMS APPROACH FOR BUSINESS, 2nd. Ed.
By NED CHAPIN, *Ph.d.*

ACCOUNTING PRINCIPLES
By ANDREW D. BRADEN, *Associate Professor of Accounting, Western Reserve University* and ROBERT G. ALLYN, *Executive Secretary, Board of C.P.A. Examiners, New York State*

MANAGERIAL ACCOUNTING
By MARY E. MURPHY, *Professor of Accounting, Los Angeles State College*

MARKET THEORY AND THE PRICE SYSTEM
By ISRAEL M. KIRZNER, *Associate Professor of Economics, New York University*

FUNDAMENTALS OF BUSINESS ENTERPRISE
By PAUL G. HASTINGS, *Associate Professor of Business Administration, Sacramento State College*

AN INTRODUCTION TO ELECTRONIC DATA PROCESSING FOR BUSINESS
By LEONARD W. HEIN, *Associate Professor of Business Administration, Los Angeles State College*

WAGE DETERMINATION: An Analysis of Wage Criteria
By JULES BACKMAN, *Professor of Economics, New York University*

INTERNATIONAL ECONOMICS
By HUGH B. KILLOUGH, *Visiting Professor of Economics, Northeastern University, and* LUCY W. KILLOUGH, *A. Barton Hepburn Professor of Economics, Wellesley College*

THE AMERICAN ECONOMY
By LEO FISHMAN, *Professor of Economics and Finance, and* BETTY G. FISHMAN, *Lecturer in Economics, both at West Virginia University*

D. VAN NOSTRAND COMPANY, INC.
120 Alexander Street Princeton, New Jersey
1532