

AGS COMPUTERS, INC.

AGS COMPUTERS
STANDARDS MANUAL

Manual of Data Processing Standards

TABLE OF CONTENTS

Revision Page

Chapter 1.	General Information
	Data Processing Standards
	Scope of Manual
	Manual: Care, Use, and Revision
	Equipment Specifications
	Software Programs
Chapter 2.	Problem Analysis and Program Development
	Problem Definition
	Program Definition
	Program Identification
	Input-Output Layout
	Card Layout
	Printer Layout
	Program Development
	Logic Design
	Coding
	Test Preparation
	Program Testing
	Documentation
Chapter 3.	Programming Standards
	Logic Design-Flow Charts
	Flow Chart Symbols
	Flow Chart Conventions
	Program Organization
	Coding Sheet Sequence
	Memory Layout
	Coding
	Character Writing Standards
	Labels
	Standard Label Chart
	Standard Halts
	Standard Messages
	Comment Cards
	Program Testing
	Test Plan Form
	Common Coding Errors
	Testing Procedure
	COBOL Programming Standards
Chapter 4.	Documentation
	Contents of Program Manual
	Disposition of Documentation and Program Cards
	Program Abstract
	Sub-Routine Documentation

AGS COMPUTERS, INC.

Chapter 5. Operating Standards
Program Library
Program Change Record

Chapter 6. Appendix
Key Punch Program Card-general
COBOL Drum Card
ALP Drum Card
FORTRAN Drum Card
Forms to be Used

AGS COMPUTERS, INC.

General Information

Revision Page

Date	By	Section	Page	Revision

AGS COMPUTERS, INC.

Chapter 1

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 programmer, 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 head of the technical staff. Suggestions may be directed to his attention 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

Equipment specifications must be obtained from the client by the lead programmer on the project. The information must answer questions in the following categories:

CPU Configuration:

model, storage, storage protect
Instruction set
Number and type of channels

I/O Devices (for each device the following is necessary):

Device type
Device Address(es)
Control unit Type and address
System Standard symbolic address
Special features (by device):
Printer: Speed, # characters
 Universal character set
Tapes: 7 and/or 9 track
 Conversion features
 Simultaneous R/W TAU
Disk units: File Scan

AGS COMPUTERS, INC.

Chapter 11

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 representative 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 adjustment 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 four-character alphanumeric field in the following format:

A B C D

Specific Module

Major component of a system

System and/or application

Major Application and/or client

The identification number is punched into card column 73-76 and appears

on the assembly listing. 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 four-character module ID is punched into the object text cards through use of the TITLE card. The module ID is also the name by which the program is known to the linkage editor through use of CSECT and START cards. Entry points to a module will be identified by a fifth character.

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 Program Planning and Testing Format for High Speed Printer, 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.

It is the responsibility of the lead programmer to establish regular access to the computer facility.

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

AGS COMPUTERS, INC.

Chapter 111

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.

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), to 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 flowchart. 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.

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 x 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, O, 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 x 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" x 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

- | | |
|-------------------|--|
| (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 31-70 | Main Logic |
| (6) Pages 71-79 | Work areas.
A separate page for each category of work area should be used. |
| (7) Page 80 | ACCUMULATORS and COUNTERS are work areas set aside in which to do arithmetic operations. |
| (8) Pages 81-89 | CONSTANTS such as numbers, edit words and alphabets to be printed in the heading. |
| Pages 90-99 | MESSAGE Work areas |

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, alphabets and special characters will be used.

Numbers

1 2 3 4 5 6 7 8 9 0

Alphabetic

A B C D E F G H I J K L M N O P Q R S T U V W X Y Z

Special Characters
(These are standard on an 029 keypunch)

¢	12-8-2	cent sign
.	12-8-3	period
L	12-8-4	less .. than
(12-8-5	open parenthesis
+	12-8-6	Plus
I	12-8-7	logical or
!	11-8-2	exclamation mark
\$	11-8-3	dollar sign
*	11-8-4	asterisk
)	11-8-5	close parenthesis
;	11-8-6	semicolon
∩	11-8-7	logical, not
∩	11	minus
/	0-1	slash

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.

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 0 (zero)
Q too easily confused with 0 (zero)
U too easily confused with V
V too easily confused with U

Labels

Labels identify program entries in coding. ALP allows for an 8 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, message it is directly related to the flow chart identification; it provides for every possible label.

Standard Label System

Character	Char.										
Z	1 Area	2	A-Accumulator C-Counter P-Punch card area				R-Read card area T-Print area W-Work area			X-Index Regis	
		3-4	Size of area; positions of core								
		5-6	Sequence number								
B	Branch point	2	Block letter from macro flow chart								
		3-4	Block number from micro flow chart								
		5-6	Sequence number								
K	Constant	2	A Address Constant	E Edit word	H Heading	L Literal	M Message	N Numeric	S Switch	T Table	X Other
		3-4	Size of constant								
		5	Sequence number	Number of decimal Sequence number	Heading line No.	Sequence Number					Size, value of constan or sequenc number
		6									
M	Message	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
Message	M	H	0	0	0	1

Area

An Area is a section of memory, defined by 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.
- SUBSCRIPT, 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.

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.

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
 - Messages, 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.

1. Exit control word shorter than field to be edited (data field)
2. Different length fields when comparing
 - Signs in units position causing incorrect compare
 - Signed fields compared to unsigned fields
3. Key punch errors: 2 and Z, 5 and S, 1 and I, 0 and Ø
4. Address arithmetic and character adjustment errors due to faulty counting or program revisions
5. Underfined symbolic operands
6. Incorrect labels on instructions causing wrong indexing in assembly
7. Constant areas overlapped with program
8. Failure to provide exit from a loop under all conditions
9. Confusion on Hi-low-equal compare
10. Arithmetic overflow
11. 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 or
object deck for test