IBM

Application Program

Documentation Aids System (1401-SE-12X)

Program Reference Manual

This system is an aid in documenting existing programs written in an assembly language for the vast majority of current systems. The system processes, on a 1401 or 1460, source programs written in SPS, Autocoder, MAP, FAP or Symbolic Flowchart Language. Source programs written in either Basic or Full Assembly Language for System/360 may also be processed. The documentation output of this system is (1) a storage map of object decks, (2) an analysis listing of source decks, and (3) a flowchart of source decks. This system provides an important new tool for documentation and conversion.

This first major section contains a general description of the system, various runs which constitute the system, machine configuration, general systems charts, a list of input/output files, and sample output. The second section, "Programmer's Information", presents program abstracts, program systems chart, general input/output description, program modification aids, system maintenance procedures, a description of the Sort Program used, and details of the Symbolic Flowchart Program. The third section contains system setup sheet, error messages and halts, system storage map, and all console operating procedures.

Copies of this and other IBM publications can be obtained through IBM branch offices. Address comments concerning the contents of this publication to IBM, Technical Publications Department, 112 East Post Road, White Plains, N. Y. 10601

CONTENTS

APPLICATION DESCRIPTION 1

APPLICATION ABSTRACT 1

SOURCE LANGUAGE 1

GENERAL SYSTEM DESCRIPTION 1
Purpose and Objectives 1
Extent of Coverage 2
Advantages 2
System Control Cards 3
Machine-Oriented Concepts 7
Control Procedures 8
Timing 8
Methods and Special Techniques 9
Restrictions 9

UPDATE PROGRAM 9
Purpose and Objectives 9
Extent of Coverage 10
Advantages 10
Update Control Cards 10
Timing 11
Restrictions and Range 11

ANALYSIS PROGRAM 11
Purpose and Objectives 11
Extent of Coverage 12
Advantages 12
Analysis Program Control Cards 13
Timing 13
Special Techniques 13
Restrictions 14

FLOWCHART PROGRAM 15
Purpose and Objectives 15
Extent of Coverage 15
Advantages 16
Flowchart Program Control Cards 16
Machine-Oriented Concepts 19
Control Procedures 19
Timing 19
Methods 20
Special Techniques 20
Restrictions and Range 22

VERIFY PROGRAM 22
Purpose and Objectives 22
Extent of Coverage 23
Advantages 23
Verify Program Control Cards 23
Control Procedures 24
Timing 24
Special Techniques 24
Restrictions 24

MACHINE AND SYSTEMS CONFIGURATION 25
Planned Use of Programming Systems 25

INPUT/OUTPUT FLOWCHARTS 25

INPUT/OUTPUT FILES 27

SAMPLE PROBLEM ANALYSIS 29
Sample Outputs 30

PROGRAMMER'S INFORMATION 38

UPDATE PROGRAM 38
Abstract 38
Description 38
System Flow 39
Input/Output Description 39

ANALYSIS PROGRAM 40
Abstract 40
Description -- Phase I 40
Description -- Phase II 49
System Flow 50
Input/Output Description 51

FLOWCHART PROGRAM 53
Abstract 53
Description -- Phase I 53
Description -- Phase II 55
System Flow 57
Input/Output Description 59
Additional Flowchart Options 61

VERIFY PROGRAM 62
Abstract 62
Description 62
System Flow 65
Input/Output Description 66

DOCUMENTATION AIDS CONTROLLER 68
Abstract 68
Resident I/O Routine Description 69
Program Selector Description (1CONA) 70
System Flow 71

DA SYSTEM MAINTENANCE PROGRAM 71
System Tape Format 72
System Maintenance Control Cards 72
Description 75
System Flow 77

PROGRAM MODIFICATION AIDS 77
General Modification Aids 77
Input/Output Modification Aids 80
Dictionary Modification Aids 82

DA SYSTEM RECORD IDENTIFICATION AND FUNCTIONS 91

APPENDIX TO PROGRAMMER'S INFORMATION 94
Sort Program 94
Symbolic Flowchart Program 96

OPERATOR'S GUIDE 103

PROGRAM SETUP 103
For DA System Operation 103
For DA System Maintenance 103

CONSOLE OPERATING INSTRUCTIONS 104

HALTS AND MESSAGE LIST 104
Operator Messages 104
Diagnostic Error Messages 106

STORAGE MAPS 112
Program Selector 112
Resident System Controller 113
Update 114
Analysis -- Phase I 115
Storage Map of Analysis -- Phase II 116
Flowcharter -- Phase I 117
Flowcharter -- Phase II 118
Verify 119
System Maintenance 120

RESTART PROCEDURES 120

BIBLIOGRAPHY 121

Form H20-0177-0
Page Revised 11/15/65
By TNL N20-0047-0

APPLICATION DESCRIPTION

APPLICATION ABSTRACT

The Documentation Aids (DA) System is designed as an aid to documenting an existing program written in an assembly language. The DA System provides machine-generated documentation aids to the vast majority of users who are programming in current IBM-supported assembly languages. The system processes programs written in Symbolic Programming System (SPS), Autocoder, Macro Assembly Program (MAP), Fortran Assembly Program (FAP), S/360 Basic Assembly Language (BAL), S/360 Full Assembly Language (FAL), or Symbolic Flowchart Language (SFL) for each of these systems:

1401/1440/1460	705/7080
1620	7040/7044
1410/7010	7090/7094
7070/7072/7074	S/360

The documentation produced by the DA System includes:

1. A storage map of object decks (except 1620, 7040/7044 and 7090/7094)
2. An analysis listing of source decks
3. A flowchart of source decks

A file maintenance program is provided as part of the DA System to aid the user in maintaining and modifying source decks.

The DA System is implemented for usage on an IBM 1401 8K, four-tape system.

SOURCE LANGUAGE

The source language used in the implementation of all DA System programs is 1401 Autocoder.

GENERAL SYSTEM DESCRIPTION

Purpose and Objectives

The DA System is designed with the following objectives:

1. To assist an installation in effectively and efficiently converting existing programs to IBM System/360 programs.
2. To encourage the user to reprogram in a higher-level language, for example, FORTRAN and COBOL.
3. To improve programming efficiency by the standardization of documentation techniques.
4. To improve and update the documentation of existing programs, thereby easing maintenance problems.
5. To eliminate many clerical and routine functions associated with documentation and conversion.
6. To provide consistent documentation for S/360 assembly language programs.

Extent of Coverage

The DA System consists of four programs:

1. Update Program allows insertion, deletion and replacement of assembly language statements in order to bring the source program up to date.

2. Analysis Program scans assembly language programs and produces pertinent information about the program scanned, including cross-references.

3. Flowchart Program (Flowcharter) scans assembly language programs and produces flowcharts of designated areas.

4. Verification Program (Verifier) produces a storage map of an object deck, noting overlay patch areas.

The use of Documentation Aids is directed towards programs written in an assembly language and processes as input either a source card deck or a tape containing card images of the source program. The Verifier processes object decks.

Programs written in the most up-to-date version (or any subset of language features of an up-to-date version) of the following assembly languages may be processed by the DA System:

SPS	Autocoder	Basic Autocoder	MAP	MAP/FAP	BAL/FAL
1401	1401/1440/1460	1401/1440/1460	7040/7044	7090/7094	S/360
1460	1410/7010	1410/7010			
1620	7070/7072/7074	7070/7072/7074			
	705/7080				

Additionally, the Flowcharter of the DA System processes programs written in Symbolic Flowchart Language (SFL). A description of SFL is given later in "Appendix to Programmer's Information".

Advantages

The advantages of using the DA System are:

1. The DA System provides a mechanized, accurate, efficient and inexpensive means of providing and maintaining up-to-date program documentation.

2. It assists every installation which is confronted to some degree by one or more of these situations:

   a. Programs seldom remain static while documentation often does. Maintenance modifications are made and application functions are added and/or deleted without updating or revising the application or program documentation.

   b. A procedure for maintaining documentation may not have been established and/or the task may not have been assigned

c. The program may have been running for a number of years and the documentation been misplaced.

d. The program may have been developed on a crash basis, and only sketchy or rough documentation developed.

e. Programmers tend to regard the documentation phase of their work as tedious and time-consuming and often neglect it unless it is demanded by project management.

3. The DA System provides an Update Program, an Analysis Program, a Flowchart Program and a Verification Program--all under the control of a System Controller.

4. All programs are integrated in a total system so that each performs certain functions which may be required by other areas.

5. The system concept enables the user to submit a source program deck to the DA System and receive any or all outputs from the system in one processing run with maximum efficiency.

6. Documentation Aids assists an installation in converting existing programs to IBM System/360 programs.

7. Input to the DA System is the original source program assembly language, either on cards or in card image form on tape. Input format (either card or tape) is determined internally by the DA System and is not specified by the user.

8. Implementation of the DA System on the IBM 1401 gives the user the opportunity to document programs for any current large-scale data processing system without tying up that system. The IBM 1401 is almost universally available, making the DA System benefits readily accessible to all users.

9. The Symbolic Flowchart Language may be used in the design and documentation of new applications.

System Control Cards

All DA processing is controlled by the use of system control cards. The form of all control cards is as follows:

Column 1	$
Columns 2 through 9	Controlling Operation
Columns 10 through 72	Operands

The operands are separated by commas. The first blank encountered in the operand field terminates the field on all but the $DAJOB card.

Notation conventions used in the description of all DA System control cards are:

1. All uppercase words are required when the functions of which they are a part are used.

2. All lowercase words represent generic terms which must be supplied by the user.

3. Material enclosed in braces, { }, indicates that a choice from the contents must be made.

4. Material enclosed in square brackets, [ ], represents an option and may be included or omitted by the user.

A brief description of the various system control cards follows (a more detailed outline of each, with operands, is given in later sections):

$DAJOB--must be the first card of each DA run; it contains the machine and language, and the program identification.

$UPDATE--calls in Update Program.

$DELETE--used with $UPDATE to indicate changes.

$ANALYZE--calls in the Analysis Program.

$CHART--calls in Flowcharter Program.

$SEGMENT--used with Chart Program to indicate areas to be charted.

$VERIFY--calls in Verification Program.

$DAEND--signified the end of a DA run.

All control cards, if present, must appear in the sequence outlined above.

The format of the $DAJOB card is:

$DAJOB {machine,} {language,} {identification}

where:

machine specifies the machine for which the source language is written, and must be one of the following:

1401	7040
1440	7040
1460	7070
1410	7072
7010	7074
1620	7090
705	7094
7080	

Additional models of the above-listed base machines are represented by the base machine number; for example, 7094 II is represented by 7094.

language — specifies the name of the language in which the source program is written and must be one of the following:

SPS
AUTO
BASIC
MAP
FAP
SFL
BAL
FAL

Where:

SPS	=	Symbolic Program System
AUTO	=	Autocoder
BASIC	=	Basic Autocoder
MAP	=	Macro Assembly Language
FAP	=	FORTRAN Assembly Language
SFL	=	Symbolic Flowchart Language
BAL	=	OS/360 Basic Assembly Language
FAL	=	OS/360 Full Assembly Language

identification — is a user-provided program identification which appears as a page heading on all DA System output reports. All columns beginning immediately after "language" through column 72 are considered as "identification".

All options must be specified in the order shown.

If the source program input is on cards, the source deck must immediately follow the $DAJOB card. If the source deck does not follow the $DAJOB card, tape input is assumed by the system.

The $DAEND card must be the last card in the input deck; its operand field is ignored.

The format of the $DAEND card is: $DAEND

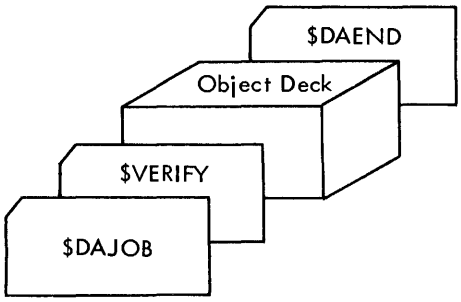
The system Controller scans the $DAJOB card to determine the machine and language to be processed. Control is passed to the program called on the next control card.

Each program in turn proceeds as requested, transferring control through the system and processing the data until the $DAEND card is reached.

In addition to the function of starting a DA run, the System Controller also provides capability for DA System maintenance.

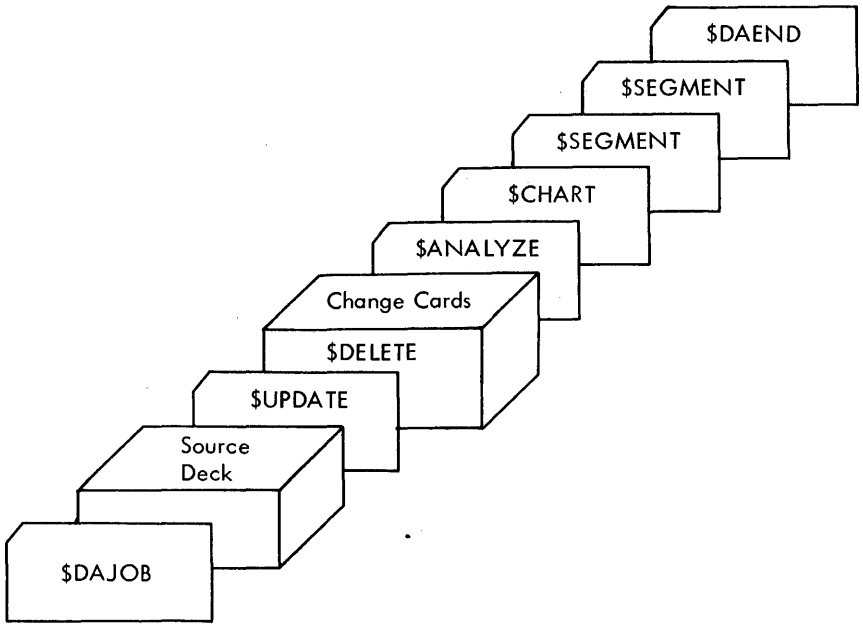
Representative Deck Set Up for Verify Program

When performing a verification run, the object deck must be supplied after the $VERIFY card, as shown:

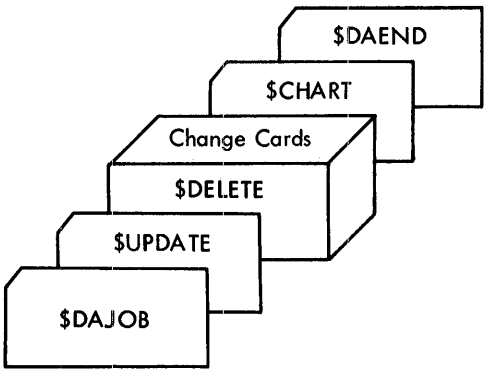


Representative Deck Set Up Using Update Program

The Update routine may be used to update a card image tape file or source deck. For the set up shown, the source language is updated and analyzed, and a flowchart is produced.



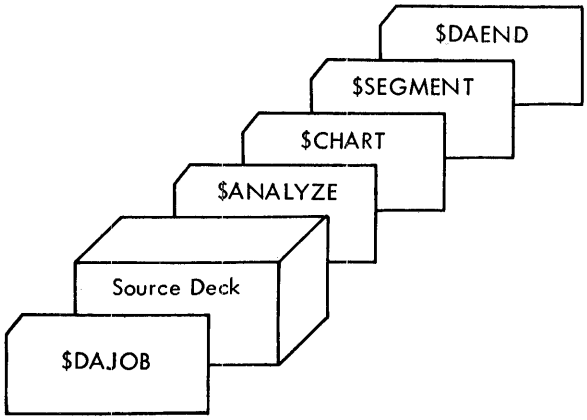
Update Analysis input. Source deck may also be in card image tape form.



Update SFL input. A flowchart is produced according to the program outlined in the updated SFL language.

Representative Deck Set Up for Analysis and Chart Programs

In this example the Analysis and Flowchart Programs are called producing analysis reports and a flowchart.



Source may also be in card-image tape form

A Chart-only run would have the same input, except that the $ANALYZE card would be omitted.

Machine-Oriented Concepts

The DA System requires four magnetic tapes for execution:

Tape Unit 1: System residence.

Tape Unit 2: Input of source language statements; intermediate storage.

Tape Unit 3: Updated source language; intermediate storage.

Tape Unit 4: Intermediate storage.

The 1402 card reader is used for three kinds of card input:

Control cards
Source language cards
Object cards.

The 1402 card punch is used for punched output.

All reports and error messages are printed on the 1403 Printer.

Control Procedures

Control cards out-of-order cause processing to terminate.

Illegal options on any control card cause processing to terminate--for example, a request to process 1401 MAP language on the $DAJOB card.

All input is checked for ascending 1401 collating sequence. All out-of-sequence conditions are noted.

System control information is supplied to the operator via the 1402 Printer.

Additional control procedures are discussed in the individual programs.

Timing

Primary considerations in estimating running time are:

1. The input medium (card or tape, tape unit model, tape density)

2. The number of statements in the source input

3. The programs called and the options specified

Approximate throughput rates are given under the timing section of the individual programs which make up the DA System. Estimates are based on the use of 729 Model V tape units at 556 cpi. However, two general timing considerations apply to the Documentation Aids System:

1. Systems Processing Overlap. Certain passes over the source deck are common to the Update, Analysis, and Flowchart Programs. If one job contains a request for any combination of the above programs, the common passes are performed only once.

2. System Tape Time. The programs are arranged on the system tape in this order:

Update Program
Analysis Program
Flowchart Program
Verify Program

The system tape passage time for any program is the sum of all system tape passage times preceding and including the program called.

Methods and Special Techniques

Specific methods and special techniques are discussed under each of the programs in the DA System. Since a large variety of languages are processed by the DA System, each language statement is scanned and converted to a standard DA System record format. Processing of the DA formatted tape in later passes is then largely language-independent.

Restrictions

The following restrictions are imposed upon the user:

1. Any unrecoverable tape errors necessitate a rerun of the job.

2. The DA System does not use or check header labels.

3. The tape input file may not be larger than the capacity of one reel of unblocked 80-character records.

4. All programs operate with single reel files only.

5. Any $ in column 1 is considered a system control card.

Additional restrictions are discussed in the individual programs.

UPDATE PROGRAM

Purpose and Objectives

The Update Program is designed to perform file maintenance on card image tapes, and is used to add or delete source statements in a program being processed by the DA System. It is also used to update the Symbolic Flowchart Language, thus providing this new language with machine maintenance capability including updated source decks and listings.

The program checks for valid sequencing and generates standard input files for other DA System programs.

Extent of Coverage

The Update Program accepts card or tape input in card image form and produces as output a card image tape. Input is checked for ascending 1401 collating sequence in columns 1-5 (SPS/Autocoder) or columns 76-80 (FAP/MAP). File maintenance is performed using the sequence field.

The user, through control card options, may request a new updated source deck and/or listing. On option, the Update Program generates ascending sequence numbers in the sequence field, starting with 00010 in increments of 10. Whenever resequencing is requested, a listing showing both old and new sequence numbers (with errors flagged) is produced.

Advantages

The advantages of using the Update Program are:

1. File maintenance is performed on source language files, including the Symbolic Flowchart Language.

2. The Update Program generates input tapes for other DA System programs.

3. All input files are checked for correct sequence.

4. Both tape and card input files are acceptable.

5. An out-of-sequence input file may be resequenced.

Update Control Cards

The Update Program is called by a $UPDATE control card.

The format of the $UPDATE card is:

$UPDATE [SEQUENCE,] [LIST,] [DECK]

where:

SEQUENCE	specifies that Update is to generate new sequence numbers in the output file. A listing with both the old and new sequence numbers is produced.
LIST	specifies that a listing of the output file is to be printed. This operand is implied if SEQUENCE is specified.
DECK	specifies that the output file is to be punched into cards.

The $UPDATE card operands may be specified in any order.

Any non-$ cards following the $UPDATE card are considered records to be added to the input file. These are merged into the input file according to their individual sequence numbers in the sequence field. Sequence errors, whether present in the input file or change file, cause processing to terminate.

To delete records from the file, the $DELETE card is used.

The format of the $DELETE card is:

$DELETE n_1 , n_2

The operands n_1 and n_2 are five-digit sequence numbers. The presence of this card in the change file causes the records between n_1 and n_2, inclusive, to be deleted from the input file. $DELETE cards are placed in the change file in sequence, with any records to be added, by their n_1 operand.

Timing

The formulas to determine the approximate running time (in seconds) for maintenance are as follows:

To generate DA tape and check sequence:

.01 x number of statements in input file

To resequence or list, add to the above:

.1 x number of statements in input file

To punch a deck and list, add:

.5 x number of statements in output file

Restrictions and Range

The following restrictions are imposed upon the user:

1. Input files must be in card image format. They may be in card form or on tape.

2. An attempt to update an out-of-sequence file causes processing to terminate after the Update run. No updated file is generated.

3. The sequence field may not contain a groupmark or a tapemark.

ANALYSIS PROGRAM

Purpose and Objectives

The Analysis Program is designed to scan an assembly language source program to provide a detailed analysis of instructions. This analysis is produced in the following forms:

1. A flagged listing denoting instruction type
2. A cross-reference dictionary of labels and references to them
3. An analysis of operation code usage.

The Analysis Program also prepares a coded assembly language tape for input to the Flowchart Program.

Extent of Coverage

The Analysis Program operates directly upon assembly language source statements and produces a flagged listing.

Flags and their indicated instruction types are:

A Assembler control
B Branch
C Complex operands
D Data defining
H Halts
I Indirect addressing
M Macros
O Input/output
R Relative addressing
X Indexed

All other instructions (for example, computational) are not flagged.

Optional reports are:

1. A frequency table of the operation codes used in the assembly language source program showing the number of times each code appears in the program.

2. A cross-reference dictionary which lists each labeled instruction and all instructions in the program which refer to that label.

Advantages

The advantages of using the Analysis Program are:

1. The flagged listing provides an up-to-date listing of the assembly language program.

2. Statements in the flagged listing are classified according to the type or nature of the statement, thus providing an aid in determining the logic flow of a program.

3. The operand references in the flagged listing provide a further aid in determining the logic flow of the program.

4. The cross-reference dictionary provides a convenient method of determining the effect of altering assembly language statements upon other portions of the program.

5. The cross-references provide a convenient method of checking for operation code and logic modification.

Analysis Program Control Cards

The Analysis Program is called in from the system tape by the $ANALYZE control card.

The format of the $ANALYZE card is:

$ANALYZE [CROSS,] [OPERAND,] [COUNT]

where:

CROSS	specifies that a cross-reference dictionary is to be printed before the flagged listing.
OPERAND	specifies that the operand references are to be included with the flagged listing.
COUNT	specifies that an operation code frequency table is to be printed.

The $ANALYZE card operands may be specified in any order.

Timing

Factors affecting the Analysis' Program processing time are:

1. The number of comment statements in source input
2. The number of references to labels in the source input program
3. The coding techniques used in the assembly language program

The formulas used to obtain approximate processing times (in seconds) are:

To produce a flagged listing:

.4 x number of assembly language statements

To produce CROSS and/or OPERAND listings:

1.5 x number of assembly language statements

Special Techniques

The following special techniques are employed by the Analysis Program:

1. Every operation code of a declarative, imperative or processor control instruction is looked up in an operation table. Associated with each operation code in the table are attribute flags which classify the type of operation. These flags are placed on the flagged listing to denote the type-of-operation code.

2. The operand field of every imperative statement is scanned to determine the nature of the statement--for example, an indexed statement, indirectly addressed statement, or a statement containing a complex operand.* Appropriate flags are generated to denote the nature of such statements.

3. For the frequency table, each operation code is looked up in the operation table, and a count is tallied of the number of times the operation code appears.

4. Records are created for symbolic operands, and tape sorts are performed to create the cross-reference dictionary and the flagged listing with operands.

5. Certain System/360 special characters (e.g., EBCDIC duals) print as blanks on the 1403 Printer. These characters are changed as follows:

			1403 Printer	
Card Punch	EBCDIC Duals	Card Punch	Chain A	Chain B
5-8	∇	4-8	@	@
12-5-8	(	0-4-8	%	(
11-5-8	)	12-4-8	¤	)
12-6-8	+	12	+	+
6-8	=	3-8	#	=

Restrictions

The following restrictions apply to Flowchart as well as Analysis output:

1. Implied indexing is not noted.

2. Operands appearing on continuation cards are not scanned.

3. Nested qualification in MAP is not analyzed. A qualifying symbol can only be up to three characters long; any excess characters are truncated. Note this can cause incorrect cross-referencing if there is more than one qualifying symbol within the source program for which the first three characters are identical.

4. Macro definitions are not entered into referencing. The operation codes within the definition appear in the Operation Code Frequency Report and the statements appear on the flagged listing, each statement flagged M.

5. With the exception of 1401 SPS and 1620 SPS, the operand field is not scanned for reference purposes or for classifying the statement if the first character of the field is blank. With the same exceptions, consecutive operands are assumed to begin in the position immediately following the operand-separating character. Therefore, for the operand

A, B

only the symbol A is recognized.

* A complex operand is defined as an operand containing any address arithmetic other than label ± constant.

6. Statements using operation codes which do not appear in the Operation Code Dictionary (for example, user-defined macros) are not scanned.

7. 1401 machine language operations beginning in column 19 are not acceptable to the DA System.

FLOWCHART PROGRAM

Purpose and Objectives

The Flowchart Program is designed to generate a flowchart of an existing source program. The flowchart produced represents the gross logic of the source program and, therefore, can be used as a guide for reprogramming in a higher-level language, for example, COBOL or FORTRAN.

The Flowchart Program scans assembly language statements which have been coded by the Analysis Program and generates a language called Symbolic Flowchart Language (SFL).

SFL is then processed producing a detailed flowchart of the original program.

Extent of Coverage

Flowcharter is logically divided into two phases. The first phase accepts as input assembly language statements which have been coded by the Analysis Program, and generates a card image tape which is used as input to the second phase.

The input instructions to the second phase, called the Symbolic Flowchart Program, constitute a language called the Symbolic Flowchart Language. This language may be used as direct input to the DA System.

Operation codes define the type of flowchart box to be generated.

Source program statements denoting input/output activity, computation, decision-making, instruction modification, subroutines, predefined processes, and logic breaks generate uniquely shaped flowchart boxes corresponding to standard flowchart conventions. Source program operands are used to insert meaningful comments into the flowchart boxes.

Labels appearing on instructions in the source program are appended to the flowchart boxes and serve as flowchart connectors as well as cross-references between the source program and the generated flowchart.

Additional cross-reference between the source program and the flowchart is provided by the sequence field.

Advantages

The advantages of using the Assembly Lanugage Flowchart Program are:

1. The shape and meaning of each flowchart box generated is consistent with the proposed American Standard, which includes all of the symbols developed by the X3.6 Committee on Flowchart Symbols for Information Processing.

2. An optional feature of Flowchart is punched output of the generated Symbolic Flowchart Language card images. By using this option, the user may manually change the logic of the flowchart or alter the comments inside the flowchart boxes simply by changing the output statements in the appropriate place. This same output, with changes, may then be resubmitted as direct input to the DA System, using SFL as the language.

3. A card image tape of the SFL language is always produced from the Flowchart. This tape may be used as input to the Update Program in subsequent passes through the DA System.

4. The assembly language statement content is reflected in the generated flowchart. In the translation from assembly language statement to symbolic language statements, labels are retained and appended to the flowchart box. Operands are retained and used to generate comments which are printed inside the flowchart box. The sequence fields are retained and printed in the flowchart box as a cross-reference between the assembly language program and the generated flowchart.

5. Flowchart examines multiple language statements, whenever possible, and combines them into a single flowchart box. Therefore, the number of generated flowchart boxes is usually substantially less than the number of assembly language statements.

Flowchart Program Control Cards

The $CHART card calls in the Assembly Language Flowchart Program from the system tape.

The format of the $CHART card is:

$CHART [DECK,] [LIST]

where:

DECK	indicates that the Flowcharter is to punch-out the program in Symbolic Flowchart Language.
LIST	indicates that the Symbolic Flowchart Language program is to be printed prior to the printing of the flowchart.

The $CHART card operands may be in either order.

The $SEGMENT card is used to segment an assembly language program. If $SEGMENT cards are used, only those statements specified are flowcharted. $SEGMENT cards are not required for small programs; however, segmentation of large programs may be required to avoid a label dictionary overflow condition within the Flowchart Program.

The format of the $SEGMENT card is:

$SEGMENT {operand 1,} {TO, / THRU,} {operand 2}

where:

operand 1	must be either a label or **. If operand 1 is a label, it is the label in the source program label field with which segmentation is to commence. If operand 1 is **, it is the first instruction of the source program and is the instruction with which segmentation is to commence.
TO	specifies that the segment terminates at, but not including, operand 2.
THRU	specifies that the segment includes and terminates with operand 2.
operand 2	signifies the end of a segment and must be either a label or **. ** is used to indicate the last statement in the source deck.

Operand 1 must precede operand 2 in the source program. If more than one segment card is used, the segments specified by the operands must appear in the same order as the assembly language program labels and not overlap.

If segmentation of an assembly language program is performed, it should be done at points which generate the fewest undefined transfer labels. Normally segmentation should be done at instructions which occur at a break in the normal logic flow--for example, ORG, EJECT.

If segmenting is performed at a label which is headed (that is, qualified by a prefix or suffix), the operand must be specified as follows:

1. For FAP and 1620 SPS, the operand consists of the heading character, followed by a dollar sign ($) and then the label. For example, if the source program is:

```
                HEAD    B
DUMP    ------
        ------
        ------
```

to specify segmenting at the DUMP symbol, the operand in the $SEGMENT card must be:

B$DUMP

If the label referred to is six characters long, the label is not headed and should appear on the $SEGMENT card without the heading character and dollar sign.

If MAP qualification is used and the heading symbol is longer than three characters, only the first three characters should be used in the $SEGMENT card operand. For example, if the source program is:

```
                QUAL    SINE
BEGIN   ------
        ------
        ------
```

the operand on the $SEGMENT card must be:

SIN$BEGIN

2. For 1410 Autocoder suffixing, the operand consists of the label, followed by as many colons as required to fill nine characters, followed by the suffixing character. For example, if the source program is:

```
                SFX     B
DUMP    ------
        ------
```

the operand in the $SEGMENT card must be:

DUMP:::::B

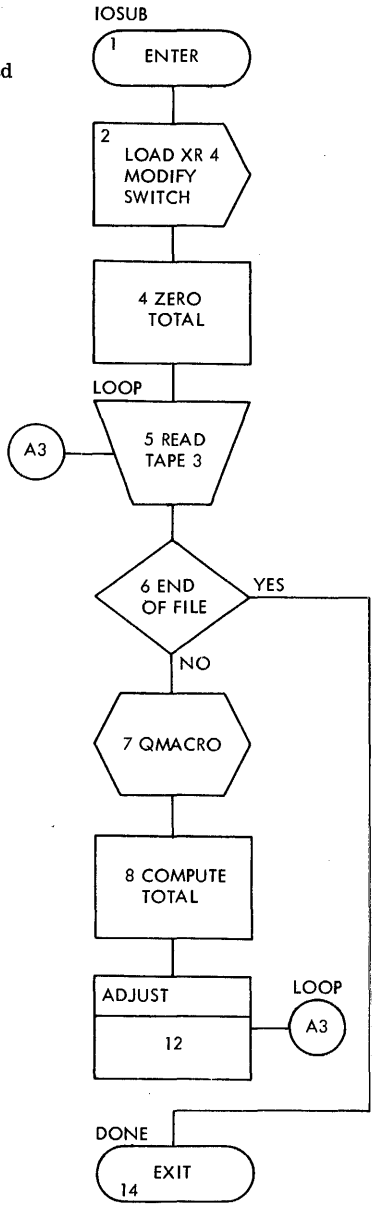
3. For 1401 Autocoder, the operand consists of the label, followed by as many colons as required to fill five characters, followed by the suffixing character. For example, using the same source program as noted in item 2, above, the operand in the $SEGMENT card must be:

DUMP:B

Phase I of Flowchart analyzes the coded assembly language statements, determines program logic, and produces SFL statements.

A detailed description of the Symbolic Flowchart Language and program is found later in "Programmer's Information". This program is utilized as Phase II of the Flowchart Program in the Documentation Aids System.

A simple MAP Assembly Language Program and the accompanying Flowcharter outputs is shown. The output of the first phase is shown in the column under Symbolic Flowchart Language, and the last column shows the flowchart generated by the second phase.



MAP Assembly Language

```
IOSUB SAVE    4
      AXT     BRANCH, 4
      SXA     SWITCH, 4
      STZ     TOTAL
LOOP  RTDB    3
      TEFB    DONE
      QMACRO  A, B
      LDQ     PRICE
      FMP     QUANTITY
      FAD     TOTAL
      STO     TOTAL
      CALL    ADJUST
      TRA     LOOP
DONE  RETURN  IOSUB
```

Symbolic Flowchart Language

```
      JOB
IOSUB ENTER1
      MODFY2 LOAD XR4 MODIFY SWITCH
      BLOCK4 ZERO TOTAL
LOOP  IO     5 READ TAPE 3
      DECID6 END OF FILE
      YES    DONE
      PREDF7 QMACRO
      BLOCK8 COMPUTE TOTAL
      SUBRTADJUST, 12
      GOTO LOOP
DONE  EXIT 14
      END
```

Machine-Oriented Concepts

Each matrix of the flowchart consists of two pages of 1403 Printer paper. The carriage control of the printer should be set at eight lines per inch. The control tape in the tape-controlled carriage should be punched in channel 1 to allow 88 lines per printer page.

A message to the operator informs that the printer carriage tape should be changed and the carriage reset for eight lines per inch.

Control Procedures

The following controls are incorporated into the Assembly Language Flowchart Program:

1. If $SEGMENT cards overlap or do not specify segmentation in the same order as the source program labels, an error message is printed and the job is terminated.

2. If the internal core capacity for procedure labels is exceeded, a message is printed informing the user that he must segment his program and the job is terminated.

Timing

In addition to those factors specified in "General System Description", certain others affect Flowcharter processing time:

1. The type of input (assembly language statements or Symbolic Flowchart Language statements).

2. The number of comment statements in source input.

3. The number of statements in the assembly language which generate flowchart boxes.

4. The coding techniques used in the assembly language program.

5. The number of labels in the source input and the number of labels generated during the derelativization process (see "Special Techniques", below).

6. The number of segment control cards in the input.

The formulas used to obtain approximate processing times (in seconds) are:

If Assembly Language input:

0.75 x number of assembly language statements

If Symbolic Flowchart Language input:

1.0 x number of Symbolic Flowchart Language statements

If an SFL deck is punched, add approximately .5 seconds per SFL card.

Methods

Conversion of assembly language statements to Symbolic Flowchart statements is performed by the following methods:

1. Procedural instructions encountered determine the type of box or connector which is generated. Only those instructions which are significant to the flowchart are used by the processor. For example, data-defining instructions are not used in the flowchart process.

2. Comments inserted in each box are based upon the types of instructions encountered.

3. Sequential procedural instructions are grouped together to generate a single flowchart box, subject to the following rules:

   a. A label appearing on a procedural instruction always causes the generation of a new box.

   b. A new box is generated whenever the assembly instructions encountered signify a change in box type.

   c. A new box is generated whenever the comments to be inserted in the current box exceed the comment capacity of the current box.

   d. A conditional branch or subroutine call instruction always generate a new box.

Special Techniques

Flowcharter employs special techniques when handling source input card images:

1. All instructions in an assembly language program which branch to a simple relative address undergo derelativization.

   To derelativize, Flowcharter computes the length of source instructions in terms of core storage positions and maintains an internal location counter.

   All source language instructions are classified as being of known or unknown length. In general, machine instructions are classified as known length and nonmachine instructions; for example, macros and pseudo operations are classified as unknown length. An internal location counter is maintained for each area of the source program of known length.

   In the following example:

   TRA label $\pm$n

   where "n" is a constant, the rule for derelativizing is as follows:

   If the location counter displacement (that is, $\pm$n) is in the same known length area as the label, the branch instruction is derelativized. If "n" is such that an unknown

length area is crossed, the branch instruction is not derelativized. In this case, Flowcharter generates a logic terminating EXIT box, rather than a GOTO connector. The same rule applies to location counter references -- for example:

BR *±n

2. Relative addresses generate labels preceded by a lozenge. Such labels do not appear on the flowchart, but are used by the program to generate connectors.

3. Branches to complex, indirect, indexed or undefined labels generate an EXIT terminal box. Undefined labels include those labels located outside the particular segment being processed.

4. All instructions in a given assembly language are classified by the type of Symbolic Flowchart Language operation they generate. Unconditional branch instructions generate a GOTO operation. Conditional branch instructions generate a DECID and a YES or NO operation. Arithmetic, logical and data movement generate a BLOCK operation.

   Instructions which modify instructions generate a MODFY operation. Instructions which are used to call subroutines generate a SUBRT operation. Instructions which define the beginning and ending of a subroutine generate an ENTER and EXIT operation, respectively. Macro instructions defined by the user generate a PREDF operation.

   Procedural operations not found in the operation table for the language specified in the $DAJOB control card generate a predefined process box.

   User-defined macros are examples of operation codes which generate predefined process boxes.

   Conditional branch instructions generate a decision box in which the condition being tested is printed in terms of hardware registers and/or fields. Three-way compare operations, such as the 7094 CAS instructions, generate two consecutive decision boxes.

   Arithmetic, internal data movement, and logical bit manipulation instructions generate a processing box. The comments generated in the box denote the general nature of the instructions encountered and, when possible, name the field stored in memory.

   Input/output operations generate an input/output box. Whenever possible, the comment printed inside the flowchart box denotes the type of operation performed (for example, READ) and names the unit, file or record acted upon.

Certain instructions in an assembly language program are calls to subroutines and generate SUBRT operations. In some assembly language programs, calls are explicitly defined by an instruction such as CALL. In other instances, calls are implied either by the instruction performing the call (for example, TSX or BTM) or by the instruction being called (for example, branch to an SBR instruction).

Restrictions and Range

1. The size of the assembly language program which can be processed is determined by the number of labels appearing on procedure instructions (not data-defining instructions) in the source program, and labels which are generated to derelativize branch instructions. If the source program is written in Symbolic Flowchart Language, there is a similar restriction on the number of labels which can be used. However, any restriction on the number of labels may be overcome by the user through proper segmentation. The number of assembly procedure labels which may be processed is 200 (plus 200 for each additional 4K of core storage). The number of SFL labels which may be processed is 390 (plus 250 for every 4K of additional core storage).

2. Only the first ten characters of assembly language fields are used in generating flowchart comments.

3. Only the first three operands of any assembly language statement are used for flowchart comments.

4. Only one (the first) operand is derelativized in branching type instructions. Exceptions to this rule are those 7070 instructions in which the second operand is the branch address (for example, BXM). For those instructions, the second operand is derelativized, if necessary.

5. Additional scanning restrictions which affect the Flowchart Program are discussed under Analysis restrictions.

VERIFY PROGRAM

Purpose and Objectives

The Verify Program is designed to help the programmer determine that the source deck is in agreement with the current object deck.

The storage map produced by the Verify Program may be compared with the original assembly listing to detect differences between the source and object programs.

Extent of Coverage

The Verify Program processes an object program deck generated by the following assembly languages:

1401 SPS
1401/1440/1460 Autocoder
1410/7010 Autocoder
705/7080 Autocoder
7070/7072/7074 Autocoder

Verifier generates a storage map and identifies overlay patches in an object program. The storage map generated represents the contents of core storage after the object program has been loaded.

Advantages

1. A detailed storage map of the object program is provided.

2. All overlay patches are identified for programmer examination.

3. Each break in location sequence is identified.

4. Verifier enables the programmer to update his source program by checking patches made to the object deck.

Verify Program Control Cards

Verifier is called in by a $VERIFY control card.

The format of the $VERIFY card is:

$VERIFY [DISK,] [LOADER]

where:

DISK — is used only when disk Autocoder is the machine language, and the format of the object program deck is condensed, containing word separator characters.

LOADER — indicates the presence of a standard loader routine in front of the object deck. Verifier recognizes the standard clear storage and bootstrap cards in the 1401/1440/1460 programs, and LOADER must be omitted in this case.

The $VERIFY card operands may be specified in any order.

The object deck immediately follows the $VERIFY card. A $DAEND control card terminates the run.

The Verify Program processes an object program in three passes.

Output from Verifier consists of the storage map; each object program instruction is printed in storage location sequence. Each printed instruction includes the storage location, the mnemonic equivalent of the operation code, the full machine language instruction and card reference number. The storage map format is comparable to the assembly listing. All overlay patches are identified by asterisks.

Control Procedures

The following control procedures are incorporated into the Verifier:

1. The object program must immediately follow a $VERIFY card in the card reader.

2. A $DAEND card signifies the end of the input object program.

Timing

The following factors affect the amount of time needed to generate a storage map of an object program:

1. Type of object program

2. Number of instructions per card

3. Number of programmed overlays and patches

A formula to determine the approximate time (in seconds) to generate a storage map is as follows:

Time = 2.0 x number of cards in object program

Special Techniques

The Verify Program employs the following special techniques in processing object programs:

1. Coded core storage addresses are converted to actual addresses.

2. A table-lookup technique is employed to determine the mnemonic equivalent of each machine operation code.

Restrictions

1. Each execute, transfer or end card signifies the end of an object program segment; therefore, overlay patches must be placed within the proper segment.

2. Data which appears to be an instruction is treated as an instruction.

3. SPS one-for-one object decks cannot be verified without condensing.

4. The LOADER option can handle only standard loaders as described in the IBM manual concerning the machine and language specified on the $DAJOB card. If a nonstandard loader is present, it should be removed from the deck.

MACHINE AND SYSTEMS CONFIGURATION

The minimum machine configuration required by the DA System is:

1. IBM 1401 or an IBM 1460 processing unit with:

   8000 positions of storage
   High-low-equal-compare
   Advanced Programming

2. IBM 1402 Card Read Punch

3. IBM 1403 Printer, Model II

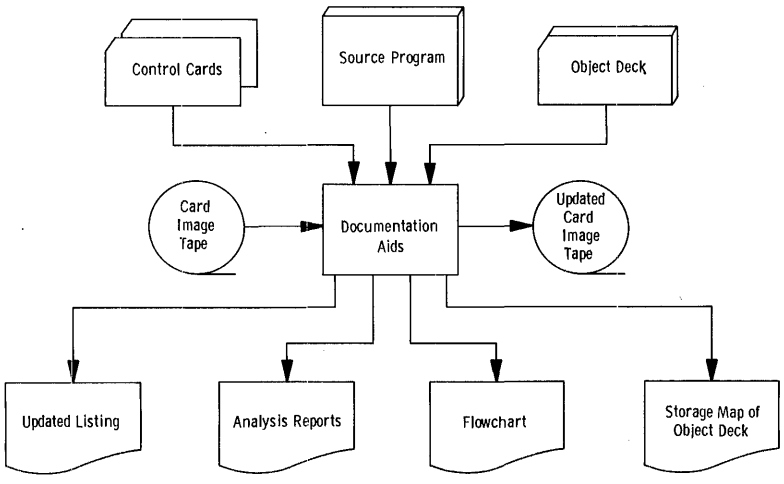
4. Four IBM 7330s or four 729 tape units, any model.

An IBM 1410 or IBM 7010 may be used when run in compatibility mode. The same minimum machine configuration as required by the IBM 1401 is applicable.

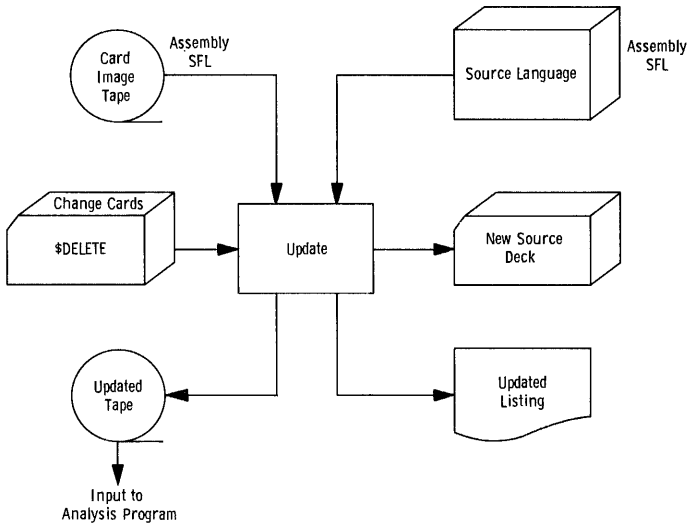
Planned Use of Programming Systems

The DA System is programmed in 1401 Autocoder language. All I/O routines and the Sort program used have been programmed internally because of the systems concept and specific requirements of this application. No other programming systems are required for implementation or modification.

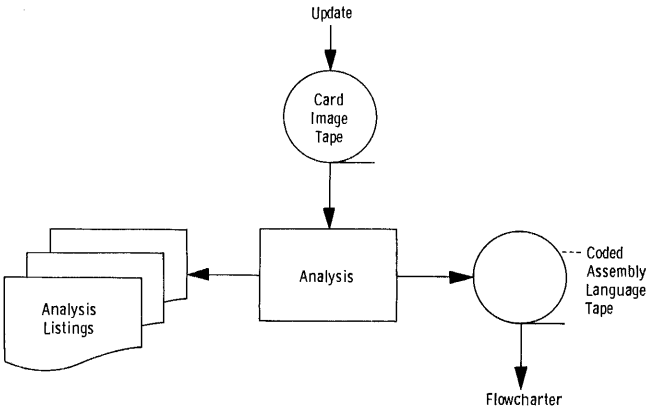
INPUT/OUTPUT FLOWCHARTS



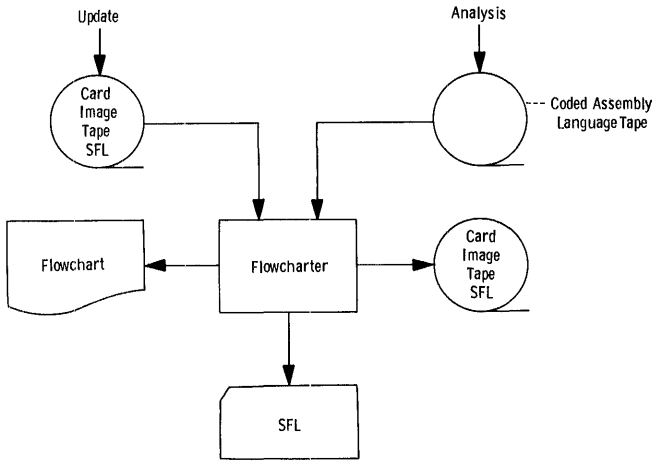
Input/output flow for DA System



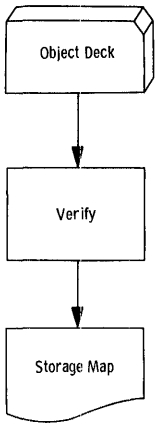
Input/output flow for update



Input/output flow for analysis



Input/output flow for flowcharter



Input/output flow for verifier

INPUT/OUTPUT FILES

The input files to the DA System are:

1. Card reader input file. This file always contains system control cards. Additionally, an assembly language program, Symbolic Flowchart Language Program, or object deck may be a part of this file.

2. Source program tape file -- unit 2. This file is used if the assembly language program or the Symbolic Flowchart Language Program resides on tape. This file must not contain system control cards. The tape format is one physical file of card images.

The output files from the Update Program are:

1. Updated source program tape file -- unit 3

2. Updated source program card file -- produced on the 1402 as a result of the DECK option.

3. Update list file -- produced on the 1403 and consisting of:

   a. A listing of the updated source program

   b. A list of all out-of-sequence conditions

   c. A list of all changes to the file

   d. Operator control messages

The output files from the Analysis Program are:

1. DA format tape file -- a coded representation of the source language.

2. Analysis reports file -- produced on the 1403 Printer and consisting of:

   a. Cross-reference report

   b. Flagged listing

   c. Operation frequency report

   d. Operator control and error messages

The output files from the Flowcharter are:

1. Card punch output file -- generated in the 1402 upon user request. This is the generated Symbolic Flowchart Language Program.

2. Symbolic flowchart language program tape file -- unit 2 -- always generated when the input is an assembly language program. The contents of this file are the same card images as the card punch output file.

3. Flowchart file -- produced on the 1403 Printer and consisting of:

   a. The flowchart

   b. A cross-reference dictionary of flowchart labels, their page and chart locations

   c. Optionally, a printout of the Symbolic Flowchart Language Program

   This file is also used to print operator control and error messages.

The output file from the Verify Program is the Storage Map File, which is produced on the 1403. Any error messages are in this file.

SAMPLE PROBLEM ANALYSIS

This section provides the user with a set of sample reports and an analysis of their use. Given a source and object deck, along with the latest assembly listing of the program, the DA System could be utilized as follows:

1. Since the DA System documents the source program, its output will be valid only insofar as the source program reflects the current running object deck. The purpose of the Verify Program is to point out any differences between the object deck generated from the original source deck and the object deck in its present status. An object deck may be altered by direct or overlay patches. The Verifier will produce a listing similar in form to the listing of assembled instructions produced by the respective system assembly program. Instructions which have two asterisks to the left of the operations code have been patched by the overlay method. The location counter of the patched instruction will indicate which characters of the preceding instruction on the Verifier listing have been affected by that patch. Note that the branch instructions at locations 862 and 899, and the add instruction at location 903 have been overlaid by a NOP instruction. Nonoverlay patches cannot be flagged, but are detected by manually matching the assembly listing and the Verify listing. The assembly and Verify listings do not match at locations 1847 through 1863, thus indicating that the constant CODE C TOTAL has been blanked out of card number 17.

2. The next step is to reflect these changes in the source program. The new symbolic entries may be manually placed in the source deck, replacing the original statements, or the Update pass of the DA System may be used. (See page containing Update Program output.) All references to an accumulation C have been eliminated. The RESEQUENCE option has provided new sequence numbers. Original statements 500 and 540 have been changed as indicated by **.

3. The updated source program is then processed through Analysis and Flowchart. Analysis produces the Operation Code Frequency Report, the Cross-Reference Report, and Flagged Listing with or without operands.

The Frequency Report gives an indication as to the general type of program by op code utilization, and may give some indication as to the conversion or reprogramming effort required.

The Cross-Reference Report is in sequence by label. Following each label are all entries which reference that label. An internally generated sequence number appears to the left of the card image. The original sequence number is shown at the right. The value of this report lies in the fact that all reference points to a given instruction, and all action taken on a given field are collected and displayed beneath the reference point in question. Line 0053 shows all usage of Index 1. The entry at line 0042 shows that the amount field is referenced by four different statements.

The Flagged Listing simplifies the logical deciphering of the program through its subreferencing of operands. Note sequence number 0012. The statement indicates a transfer to ADDA if CODE + X1 is an A. Taken in union with the subreferences, one sees that CODE is a subfield to a DA statement, and at ADDA the amount field will be added to WORKA. The format of this report is similar to that of the Cross-Reference Report, with the addition of coded flags to the left of the internal sequence number.

Pages 15 and 16 of the sample problem provide the Cross-Reference and Label Dictionaries produced by Flowchart. Label ADDA appears at matrix position B0 on the flowchart and reference to it is at A7. ⌶ 3000 is a generated label to provide linkage connection between B3 and A6.

Page 17 of the sample problem represents the DA System flowchart of the program. In each flowchart box is a sequence number, by means of which the flowchart, Flagged Listing and Cross-Reference Reports may be coordinated to effect complete documentation of the entire program.

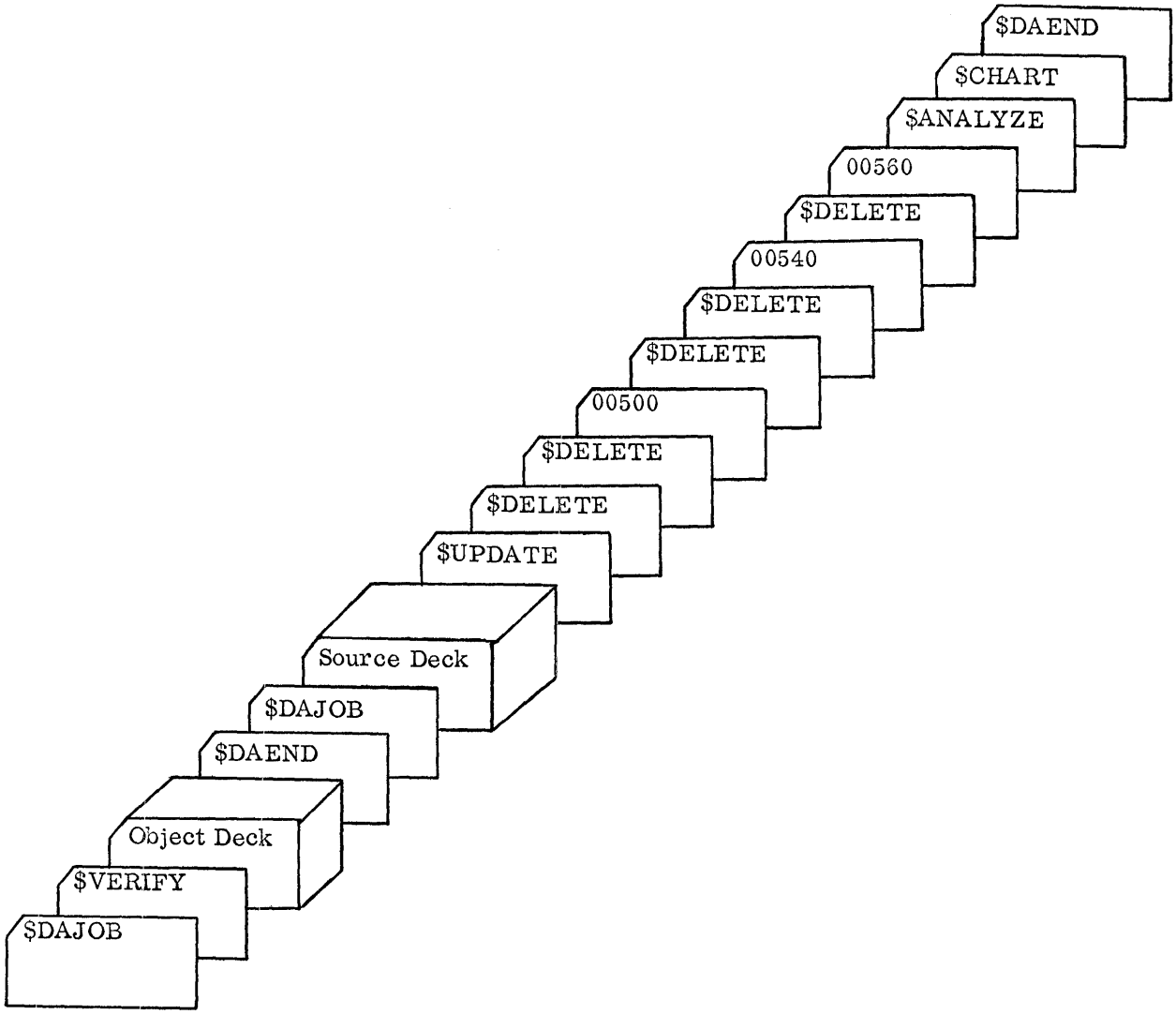
Sample Outputs

The following pages reflect sample output data for the DA System.

Autocoder Listing

```
CLEAR STORAGE 1     ,008015,022026,030037,044,049,053053N000000NC0001026                                    1
CLEAR STORAGE 2     L068116,105106,1101178101/19Z#071029C029056R026/B001/0991,001/001117106                  2
BOOTSTRAP           ,008015,022029,036040,047054,061068,072/061039              ,0010011040                 3

                              DOCUMENTATION AIDS SAMPLE PROBLEM                                    PAGE    1

SEQ PG LIN  LABEL   OP   OPERANDS                                  SFX CT  LOCN  INSTRUCTION TYPE  CARD

101    010  000     JOB  DOCUMENTATION AIDS SAMPLE PROBLEM
102    020          CTL  441
103    030          ORG  800                  START ASSEMBLY AT 800          0800
104    040  START   BLC  END                  ON LAST CARD GO TO END       5  0800  B 947 A              4
105    050          R                         READ A CARD                  1  0805  1                    4
106    060          MCM  1,OUTPUT&X1          TRANSFER CARD TO OUTPUT AREA 7  0806  P 001 9Z0             4
107    070  MODIFY  MA   @081@,X1             UP INDEX X1 BY THE RECORD LENGTH 7  0813  # Z22 089        4
108    080          BCE  *&5,X1,0             10 TIMES 81 EQUALS 810       8  0820  B 832 089 0          4
109    090          B    START                GO READ ANOTHER RECORD       4  0828  B 800               4
110    100  ADD     SBR  X1,0                 ZERO INDEX 1                 7  0832  H 089 000            4
111    110          A    AMOUNT&X1,TOTAL      ACCUMULATE OVERALL TOTAL     7  0839  A 9Z9 Z19            5
112    120          BCE  ADDA,CODE&X1,A       IF MEMBER OF CLASS A         8  0846  B 881 9Z0 A          5
113    130          BCE  ADDB,CODE&X1,B       IF MEMBER OF CLASS B         8  0854  B 892 9Z0 B          5
114    140          BCE  ADDC,CODE&X1,C       IF MEMBER OF CLASS C         8  0862  B 903 9Z0 C          5
115    150          A    AMOUNT&X1,OTHER      ACCUM ALL OTHER CLASSES      7  0870  A 9Z9 Y94            5
116    160          B    UPX1                 TRANSFER TO STEP-UP INDEX 1  4  0877  B 910               6
117    170  ADDA    A    AMOUNT&X1,WORKA      ACCUMULATE A-CLASS           7  0881  A 9Z9 Y22            6
118    180          B    UPX1                 TRANSFER TO STEP-UP X1       4  0888  B 910               6
119    190  ADDB    A    AMOUNT-2&X1,WORKB    ACCUMULATE B-CLASS           7  0892  A 9Z7 Y46            6
120    200          B    *&8                  TRANSFER TO STEP-UP X1       4  0899  B 910               6
121    220  ADDC    A    AMOUNT&X1,WORKC      ACCUMULATE C-CLASS           7  0903  A 9Z9 Y71            6
122    220  UPX1    MA   @081@,X1             STEP-UP X1 FOR NEXT RECORD   7  0910  # Z22 089        7
123    230          BCE  WRITE1,X1,0          GO WRITE THE BLOCK IF X1   810  8  0917  B 929 089 0          7
124    240          B    ADD&7                GO TO ACCUM FROM NEXT RECORD 4  0925  B 839               7
125    250  WRITE1  B    WRITE                GO TO WRITE AND RETURN TO NSI 4  0929  B 400               7
126    260          DCW  &OUTPUT              ADDRESS OF OUTPUT AREA       3  0935  990                  7
127    270  WRITE   EQU  400        ADDRESS OF PRECOMPILED WRITE ROUTINE        0400
128    280          SBR  X1,0                 ZERO INDEX 1                 7  0936  H 089 000            7
129    290          B    START                GO TO READ 10 CARDS          4  0943  B 800               7
130    300  END     B    CLOSE                GO TO CLOSE THE OUTPUT FILE  4  0947  B 500               8
131    310  CLOSE   EQU  500        ADDRESS OF PRECOMPILED CLOSE ROUTINE        0500
132    320          WTM  2                                                 5  0951  U %U2 M              8
133    330          RWU  2                    REWIND & UNLOAD OUTPUT FILE  5  0956  U %U2 U              8
134    340          MCM  WORK,201             MOVE ALL TOTALS TO PRINT AREA 7  0961  P Y01 201            8
135    350          CC   A                                                2  0968  F A                 8
136    360          W                                                      1  0970  2                    8
137    370          CS   320                  CLEAR THE PRINT AREA         4  0971  / 320               8
138    380          CS                                                     1  0975  /                    9
139    390  END1    MLC  @END OF JOB@,250     MOVE EOJ MESSAGE TO PRINT    7  0976  M Z32 250            9
140    400          CC   A                                                2  0983  F A                 9
141    410          W                                                      1  0985  2                    9
142    420          H    END1                 FINAL HALT                   4  0986  . 976                9
143    430  OUTPUT  DA   10X81,G              OUTPUT AREA                     0990   1799                 11
144    440  CODE    1,1                                                     0990          FIELD   11
145    450  AMOUNT  2,10                                                    0999          FIELD   13
146    460  NAME    11,31                                                   1020          FIELD   14
               DCW  @ @                                                 1  1800          GMARK   16

                              DOCUMENTATION AIDS SAMPLE PROBLEM                                    PAGE    2

SEQ PG LIN  LABEL   OP   OPERANDS                                  SFX CT  LOCN  INSTRUCTION TYPE  CARD

147    470  WORK    EQU  *&1                                                1801
148    480          DCW  @CODE A TOTAL #@                           14  1814                        16
149    490  WORKA   DCW  #8                                          8  1822                        16
150    500          DCW  @  CODE B TOTAL  @                         16  1838                        16
151    510  WORKB   DCW  #8                                          8  1846                        17
152    520          DCW  @  CODE C TOTAL  @                         17  1863                        17
153    530  WORKC   DCW  #8                                          8  1871                        17
154    540          DCW  @  OTHER TOTAL  @                          15  1886                        18
155    550  OTHER   DCW  #8                                          8  1894                        18
156    560          DCW  @  GRAND TOTAL  @                          15  1909                        18
157    570  TOTAL   DCW  #10                                        10  1919                        19
158    580  X1      EQU  89                   DE                            0089
               DCW  @081@                                           3  1922          LIT     19
         139        @END OF JOB@                                    10  1932          LIT     19
159    590          END  START                                             / 800 080               20
```

DA System Sample Problem Deck Setup



Output of Verify Run

```
$VERIFY          DOCUMENTATION AIDS SAMPLE PROBLEM                PAGE    3

OP       CT       LOCN  INSTRUCTION      CARD

BLC       5        800  B 947 A             4
R         1        805  1                   4
MCM       7        806  P 001 920           4
MA        7        813  # Z22 089           4
BCE       8        820  B 832 089 0         4
B         4        828  B 800               4
SBR       7        832  H 089 000           4
A         7        839  A 929 Z19           5
BCE       8        846  B 881 920 A         5
BCE       8        854  B 892 920 B         5
BCE       8        862  B 903 920 C         5
NOP       1    **  862  N
A         7        870  A 929 Y94           5
B         4        877  B 910               6
A         7        881  A 929 Y22           6
B         4        888  B 910               6
A         7        892  A 927 Y46           6
B         4        899  B 910               6
NOP       1    **  899  N
A         7        903  A 929 Y71           6
NOP       1    **  903  N
MA        7        910  # Z22 089           7
BCE       8        917  B 929 089 0         7
B         4        925  B 839               7
B         4        929  B 400               7
          3        933  990                 7
SBR       7        936  H 089 000           7
B         4        943  B 800               7
B         4        947  B 500               8
U*I/O*    5        951  U %U2 M             8
U*I/O*    5        956  U %U2 U             8
MCM       7        961  P Y01 201           8
CC   A    2        968  F A                 8
W         1        970  2                   8
CS        4        971  / 320               8
CS        1        975  /                   9
MLC       7        976  M Z32 250           9
CC   A    2        983  F A                 9
W         1        985  2                   9
H         4        986  . 976               9
                   990                      9
                   990                     11
                   991                     13
                  1000                     14
                  1071                      9
                  1071                     11
                  1072                     13
                  1081                     14
                  1152                     10
                  1152                     11
                  1153                     13
                  1162                     15
                  1233                     10
```

```
$VERIFY          DOCUMENTATION AIDS SAMPLE PROBLEM                PAGE    4

OP       CT       LOCN  INSTRUCTION      CARD

                  1233                     11
                  1234                     13
                  1243                     15
                  1314                     10
                  1314                     12
                  1315                     13
                  1324                     15
                  1395                     10
                  1395                     12
                  1396                     13
                  1405                     15
                  1476                     10
                  1476                     12
                  1477                     14
                  1486                     15
                  1557                     10
                  1557                     12
                  1558                     14
                  1567                     15
                  1638                     11
                  1638                     12
                  1639                     14
                  1648                     16
                  1719                     11
                  1719                     12
                  1720                     14
                  1729                     16
          1       1800  GMWM               16
         12       1801  CODE A TOTAL       16
          2       1813   #                 16
          8       1815                     16
         12       1823     CODE B TOT      16
          4       1835  AL                 16
MA        1    **  1838  #
          8       1839                     17
         12       1847                     17
          5       1859                     17
          8       1864                     17
         12       1872     OTHER TOTA      18
          3       1884  L                  18
MA        1    **  1886  #
          8       1887                     18
         12       1895     GRAND TOTA      18
          3       1907  L                  18
MA        1    **  1909  #
         10       1910                     19
          3       1920  OH1                19
         10       1923  END OF JOB         19
                        / 800 080          20
```

Form H20-0177-0
Page Revised 11/15/65
By TNL N20-0047-0

Output of Documentation Run

```
$UPDATE          DOCUMENTATION AIDS SAMPLE PROBLEM                          PAGE    3

00010          JOB  DOCUMENTATION AIDS SAMPLE PROBLEM                        00010
00020          CTL  441                                                      00020
00030          ORG  800                START ASSEMBLY AT 800                 00030
00040START     BLC  END                ON LAST CARD GO TO END                00040
00050          R                       READ A CARD                           00050
00060          MCM  1,OUTPUT&X1        TRANSFER CARD TO OUTPUT AREA          00060
00070MODIFY    MA   @081@,X1           UP INDEX 1 BY THE RECORD LENGTH       00070
00080          BCE  *&5,X1,0           10 TIMES 81 EQUALS 810                00080
00090          B    START              GO READ ANOTHER RECORD                00090
00100ADD       SBR  X1,0               ZERO INDEX 1                          00100
00110          A    AMOUNT&X1,TOTAL    ACCUMULATE OVERALL TOTAL              00110
00120          BCE  ADDA,CODE&X1,A     IF MEMBER OF CLASS A                  00120
00130          BCE  ADDB,CODE&X1,B     IF MEMBER OF CLASS B                  00130
                                                                             DELETE  00140,00140
00140          A    AMOUNT&X1,OTHER    ACCUM ALL OTHER CLASSES               00150
00150          B    UPX1               TRANSFER TO STEP-UP INDEX 1           00160
00160ADDA      A    AMOUNT&X1,WORKA    ACCUMULATE A-CLASS                    00170
00170          B    UPX1               TRANSFER TO STEP-UP X1                00180
00180ADDB      A    AMOUNT-2&X1,WORKB  ACCUMULATE B-CLASS                    00190
                                                                             DELETE  00200,00220
00190UPX1      MA   @081@,X1           STEP-UP X1 FOR NEXT RECORD            00225
00200          BCE  WRITE1,X1,0        GO WRITE THE BLOCK IF X1   810        00230
00210          B    ADD&7              GO TO ACCUM FROM NEXT RECORD          00240
00220WRITE1    B    WRITE              GO TO WRITE AND RETURN TO NSI         00250
00230          DCW  &OUTPUT            ADDRESS OF OUTPUT AREA                00260
00240WRITE     EQU  400        ADDRESS OF PRECOMPILED WRITE ROUTINE          00270
00250          SBR  X1,0               ZERO INDEX 1                          00280
00260          B    START              GO TO READ 10 CARDS                   00290
00270END       B    CLOSE              GO TO CLOSE THE OUTPUT FILE           00300
00280CLOSE     EQU  500        ADDRESS OF PRECOMPILED CLOSE ROUTINE          00310
00290          WTM  2                                                        00320
00300          RWU  2                  REWIND & UNLOAD OUTPUT FILE           00330
00310          MCM  WORK,201           MOVE ALL TOTALS TO PRINT AREA         00340
00320          CC   A                                                        00350
00330          W                                                             00360
00340          CS   320                CLEAR THE PRINT AREA                  00370
00350          CS                                                            00380
00360END1      MLC  @END OF JOB@,250   MOVE EOJ MESSAGE TO PRINT             00390
00370          CC   A                                                        00400
00380          W                                                             00410
00390          H    END1               FINAL HALT                            00420
00400OUTPUT    DA   10X81,G            OUTPUT AREA                           00430
00410CODF          1,1                                                       00440
00420AMOUNT        2,10                                                      00450
00430NAME          11,31                                                     00460
00440WORK      EQU  *&1                                                      00470
00450          DCW  @CODE A TOTAL #@                                         00480
00460WORKA     DCW  #8                                                       00490
                                                                             DELETE  00500,00500
00470          DCW  @  CODE B TOTAL #@                                      **00500
00480WORKB     DCW  #8                                                       00510
                                                                             DELETE  00520,00530
                                                                             DELETE  00540,00540
00490          DCW  @  OTHER TOTAL #@                                       **00540
00500OTHER     DCW  #8                                                       00550
```

```
$UPDATE          DOCUMENTATION AIDS SAMPLE PROBLEM                          PAGE    4

                                                                             DELETE  00560,00560
00510          DCW  @  GRAND TOTAL #@                                       **00560
00520TOTAL     DCW  #10                                                      00570
00530X1        EQU  89                 DE                                    00580
00540          END  START                                                    00590
```

```
ANALYSIS         DOCUMENTATION AIDS SAMPLE PROBLEM                          PAGE  7

     OPERATION CODE FREQUENCY REPORT

      MNEMONICS      TALLY

        A              4
        B              7
        BCE            4
        BLC            1
        CC             2
        CS             2
        CTL            1
        DA             4
        DCW            9
        END            1
        EQU            4
        H              1
        JOB            1
        MA             2
        MCM            2
        MLC            1
        ORG            1
        R              1
        RWU            1
        SBR            2
        W              2
        WTM            1

                 TYPE        TOTAL    PERCENT

              INPUT-OUTPUT     007      .13
              DATA DEFINING    013      .24
              BRANCH           012      .22
              HALT             001      .02
              ASSEMBLER        008      .15
              OTHER            013      .24

          TOTAL       54
```

```
ANALYSIS        DOCUMENTATION AIDS SAMPLE PROBLEM                    PAGE  8

                                CROSS REFERENCE REPORT

0010            ADD     SBR  X1,0                ZERO INDEX 1                                    00100
     0021                    B    ADD&7                GO TO ACCUM FROM NEXT RECORD                    00210
0016            ADDA    A    AMOUNT&X1,WORKA      ACCUMULATE A-CLASS                              00160
     0012                    BCE  ADDA,CODE&X1,A       IF MEMBER OF CLASS A                            00120
0018            ADDB    A    AMOUNT-2&X1,WORKB    ACCUMULATE B-CLASS                              00180
     0013                    BCE  ADDB,CODE&X1,B       IF MEMBER OF CLASS B                            00130
0042            AMOUNT  DC   2,10                                                                 00420
     0011                    A    AMOUNT&X1,TOTAL      ACCUMULATE OVERALL TOTAL                        00110
     0014                    A    AMOUNT&X1,OTHER      ACCUM ALL OTHER CLASSES                         00140
     0016                ADDA A    AMOUNT&X1,WORKA      ACCUMULATE A-CLASS                              00160
     0018                ADDB A    AMOUNT-2&X1,WORKB    ACCUMULATE B-CLASS                              00180
0028            CLOSE   EQU  500     ADDRESS OF PRECOMPILED CLOSE ROUTINE                            00280
     0027                    END  B    CLOSE                GO TO CLOSE THE OUTPUT FILE                     00270
0041            CODE    DC   1,1                                                                    00410
     0012                    BCE  ADDA,CODE&X1,A       IF MEMBER OF CLASS A                            00120
     0013                    BCE  ADDB,CODE&X1,B       IF MEMBER OF CLASS B                            00130
0027            END     B    CLOSE                GO TO CLOSE THE OUTPUT FILE                     00270
     0004                    START BLC  END                ON LAST CARD GO TO END                          00040
0036            END1    MLC  @END OF JOB@,250     MOVE EOJ MESSAGE TO PRINT                       00360
     0039                    H    END1                 FINAL HALT                                      00390
0007            MODIFY  MA   @081@,X1                UP INDEX 1 BY THE RECORD LENGTH                 00070
0043            NAME    DC   11,31                                                                  00430
0050            OTHER   DCW  #8                                                                     00500
     0014                    A    AMOUNT&X1,OTHER      ACCUM ALL OTHER CLASSES                         00140
0040            OUTPUT  DA   10X81,G                  OUTPUT AREA                                     00400
     0006                    MCM  1,OUTPUT&X1          TRANSFER CARD TO OUTPUT AREA                    00060
     0023                    DCW  &OUTPUT              ADDRESS OF OUTPUT AREA                          00230
0004            START   BLC  END                  ON LAST CARD GO TO END                          00040
     0026                    B    START                GO TO READ 10 CARDS                             00260
     0054                    END  START                                                               00540
     0009                    B    START                GO READ ANOTHER RECORD                          00090
0052            TOTAL   DCW  #10                                                                    00520
     0011                    A    AMOUNT&X1,TOTAL      ACCUMULATE OVERALL TOTAL                        00110
0019            UPX1    MA   @081@,X1                STEP-UP X1 FOR NEXT RECORD                      00190
     0015                    B    UPX1                 TRANSFER TO STEP-UP INDEX 1                     00150
     0017                    B    UPX1                 TRANSFER TO STEP-UP X1                          00170
0044            WORK    EQU  *&1                                                                    00440
     0031                    MCM  WORK,201             MOVE ALL TOTALS TO PRINT AREA                   00310
0046            WORKA   DCW  #8                                                                     00460
     0016                ADDA A    AMOUNT&X1,WORKA      ACCUMULATE A-CLASS                              00160
0048            WORKB   DCW  #8                                                                     00480
     0018                ADDB A    AMOUNT-2&X1,WORKB    ACCUMULATE B-CLASS                              00180
0024            WRITE   EQU  400     ADDRESS OF PRECOMPILED WRITE ROUTINE                            00240
     0022                    WRITE1 B    WRITE                GO TO WRITE AND RETURN TO NSI                   00220
0022            WRITE1  B    WRITE                GO TO WRITE AND RETURN TO NSI                   00220
     0020                    BCE  WRITE1,X1,0          GO WRITE THE BLOCK IF X1   810                  00200
0053            X1      EQU  89                   DE                                              00530
     0007                    MODIFY MA   @081@,X1          UP INDEX 1 BY THE RECORD LENGTH                 00070
     0008                    BCE  *&5,X1,0             10 TIMES 81 EQUALS 810                          00080
     0010                    ADD  SBR  X1,0            ZERO INDEX 1                                    00100
     0019                    UPX1 MA   @081@,X1        STEP-UP X1 FOR NEXT RECORD                      00190
```

```
ANALYSIS        DOCUMENTATION AIDS SAMPLE PROBLEM                    PAGE  9

                                CROSS REFERENCE REPORT

0025                    SBR  X1,0                ZERO INDEX 1                                    00250
0006                    MCM  1,OUTPUT&X1          TRANSFER CARD TO OUTPUT AREA                    00060
0011                    A    AMOUNT&X1,TOTAL      ACCUMULATE OVERALL TOTAL                        00110
0014                    A    AMOUNT&X1,OTHER      ACCUM ALL OTHER CLASSES                         00140
0016                ADDA A    AMOUNT&X1,WORKA      ACCUMULATE A-CLASS                              00160
0018                ADDB A    AMOUNT-2&X1,WORKB    ACCUMULATE B-CLASS                              00180
0020                    BCE  WRITE1,X1,0          GO WRITE THE BLOCK IF X1   810                  00200
0012                    BCE  ADDA,CODE&X1,A       IF MEMBER OF CLASS A                            00120
0013                    BCE  ADDB,CODE&X1,B       IF MEMBER OF CLASS B                            00130
```

FLAGGED LISTING

```
A       0001                    JOB  DOCUMENTATION AIDS SAMPLE PROBLEM                                    00010
A       0002                    CTL  441                                                                  00020
A       0003                    ORG  800              START ASSEMBLY AT 800                               00030
B       0004          START     BLC  END              ON LAST CARD GO TO END                              00040
                0027            END      B    CLOSE             GO TO CLOSE THE OUTPUT FILE                    00270
0       0005                    R                  READ A CARD                                            00050
X       0006                    MCM  1,OUTPUT&X1       TRANSFER CARD TO OUTPUT AREA                       00060
                0040            OUTPUT   DA   10X81,G           OUTPUT AREA                                    00400
                0053            X1       EQU  89                DE                                             00530
                0007    MODIFY  MA   @081@,X1          UP INDEX 1 BY THE RECORD LENGTH                    00070
                0053            X1       EQU  89                DE                                             00530
R,B     0008                    BCE  *65,X1,0          10 TIMES 81 EQUALS 810                             00080
                0053            X1       EQU  89                DE                                             00530
B       0009                    B    START             GO READ ANOTHER RECORD                             00090
                0004            START    BLC  END               ON LAST CARD GO TO END                         00040
                0010    ADD     SBR  X1,0              ZERO INDEX 1                                       00100
                0053            X1       EQU  89                DE                                             00530
X       0011                    A    AMOUNT&X1,TOTAL   ACCUMULATE OVERALL TOTAL                            00110
                0042            AMOUNT   2,10                                                                  00420
                0053            X1       EQU  89                DE                                             00530
                0052            TOTAL    DCW  #10                                                              00520
X,B     0012                    BCE  ADDA,CODE&X1,A    IF MEMBER OF CLASS A                               00120
                0016            ADDA     A    AMOUNT&X1,WORKA      ACCUMULATE A-CLASS                        00160
                0041            CODE     1,1                                                                   00410
                0053            X1       EQU  89                DE                                             00530
X,B     0013                    BCE  ADDB,CODE&X1,B    IF MEMBER OF CLASS B                               00130
                0018            ADDB     A    AMOUNT-2&X1,WORKB    ACCUMULATE B-CLASS                        00180
                0041            CODE     1,1                                                                   00410
                0053            X1       EQU  89                DE                                             00530
X       0014                    A    AMOUNT&X1,OTHER   ACCUM ALL OTHER CLASSES                             00140
                0042            AMOUNT   2,10                                                                  00420
                0053            X1       EQU  89                DE                                             00530
                0050            OTHER    DCW  #8                                                               00500
B       0015                    B    UPX1              TRANSFER TO STEP-UP INDEX 1                        00150
                0019            UPX1     MA   @081@,X1          STEP-UP X1 FOR NEXT RECORD                    00190
X       0016    ADDA            A    AMOUNT&X1,WORKA   ACCUMULATE A-CLASS                                 00160
                0042            AMOUNT   2,10                                                                  00420
                0053            X1       EQU  89                DE                                             00530
                0046            WORKA    DCW  #8                                                               00460
B       0017                    B    UPX1              TRANSFER TO STEP-UP X1                             00170
                0019            UPX1     MA   @081@,X1          STEP-UP X1 FOR NEXT RECORD                    00190
R,X     0018    ADDB            A    AMOUNT-2&X1,WORKB    ACCUMULATE B-CLASS                            00180
                0042            AMOUNT   2,10                                                                  00420
                0053            X1       EQU  89                DE                                             00530
                0048            WORKB    DCW  #8                                                               00480
                0019    UPX1    MA   @081@,X1          STEP-UP X1 FOR NEXT RECORD                         00190
                0053            X1       EQU  89                DE                                             00530
B       0020                    BCE  WRITE1,X1,0       GO WRITE THE BLOCK IF X1   810                     00200
                0022            WRITE1   B    WRITE             GO TO WRITE AND RETURN TO NSI                  00220
                0053            X1       EQU  89                DE                                             00530
R,B     0021                    B    ADD&7             GO TO ACCUM FROM NEXT RECORD                       00210
                0010            ADD      SBR  X1,0              ZERO INDEX 1                                   00100
```


FLAGGED LISTING

```
B       0022    WRITE1  B    WRITE             GO TO WRITE AND RETURN TO NSI                      00220
                0024            WRITE    EQU  400       ADDRESS OF PRECOMPILED WRITE ROUTINE               00240
D       0023                    DCW  &OUTPUT           ADDRESS OF OUTPUT AREA                             00230
                0040            OUTPUT   DA   10X81,G           OUTPUT AREA                                    00400
A       0024    WRITE   EQU  400       ADDRESS OF PRECOMPILED WRITE ROUTINE                               00240
                0025            SBR  X1,0              ZERO INDEX 1                                       00250
                0053            X1       EQU  89                DE                                             00530
B       0026                    B    START             GO TO READ 10 CARDS                                00260
                0004            START    BLC  END               ON LAST CARD GO TO END                         00040
B       0027    END     B    CLOSE             GO TO CLOSE THE OUTPUT FILE                        00270
                0028            CLOSE    EQU  500       ADDRESS OF PRECOMPILED CLOSE ROUTINE               00280
A       0028    CLOSE   EQU  500       ADDRESS OF PRECOMPILED CLOSE ROUTINE                               00280
0       0029                    WTM  2                                                                    00290
0       0030                    RWU  2                 REWIND & UNLOAD OUTPUT FILE                        00300
                0031            MCM  WORK,201          MOVE ALL TOTALS TO PRINT AREA                      00310
                0044            WORK     EQU  *&1                                                              00440
0       0032                    CC   A                                                                    00320
0       0033                    W                                                                         00330
                0034            CS   320               CLEAR THE PRINT AREA                               00340
                0035            CS                                                                        00350
                0036    END1    MLC  @END OF JOB@,250  MOVE EOJ MESSAGE TO PRINT                          00360
0       0037                    CC   A                                                                    00370
0       0038                    W                                                                         00380
H       0039                    H    END1              FINAL HALT                                         00390
                0036            END1     MLC  @END OF JOB@,250  MOVE EOJ MESSAGE TO PRINT                  00360
D       0040    OUTPUT  DA   10X81,G           OUTPUT AREA                                                00400
D       0041    CODE    1,1                                                                               00410
D       0042    AMOUNT  2,10                                                                              00420
D       0043    NAME    11,31                                                                             00430
R,A     0044    WORK    EQU  *&1                                                                          00440
D       0045            DCW  @CODE A TOTAL #@                                                             00450
D       0046    WORKA   DCW  #8                                                                           00460
D       0047            DCW  @  CODE B TOTAL #@                                                           00470
D       0048    WORKB   DCW  #8                                                                           00480
D       0049            DCW  @  OTHER TOTAL #@                                                            00490
D       0050    OTHER   DCW  #8                                                                           00500
D       0051            DCW  @  GRAND TOTAL #@                                                            00510
D       0052    TOTAL   DCW  #10                                                                          00520
A       0053    X1      EQU  89                DE                                                         00530
A       0054            END  START                                                                        00540
                0004            START    BLC  END               ON LAST CARD GO TO END                         00040
```

```
        CHART              DOCUMENTATION AIDS SAMPLE PROBLEM                    PAGE  14
                                    SEGMENT **, THRU, **

LABEL       DEFINED AT

¤30000          1 A6
ADD             1 A5
ADDA            1 B0
ADDB            1 B1
END             1 B6
END1            1 C1
MODIFY          1 A3
START           1 A0
UPX1            1 B2
WRITE1          1 B4
```

```
        CHART              DOCUMENTATION AIDS SAMPLE PROBLEM                    PAGE  15
                                    SEGMENT **, THRU, **

LABEL       REFERENCES

¤30000          1 B3
ADDA            1 A7
ADDB            1 A8
END             1 A0
END1            1 C3
START           1 A4        1 B5
UPX1            1 A9        1 B0
```

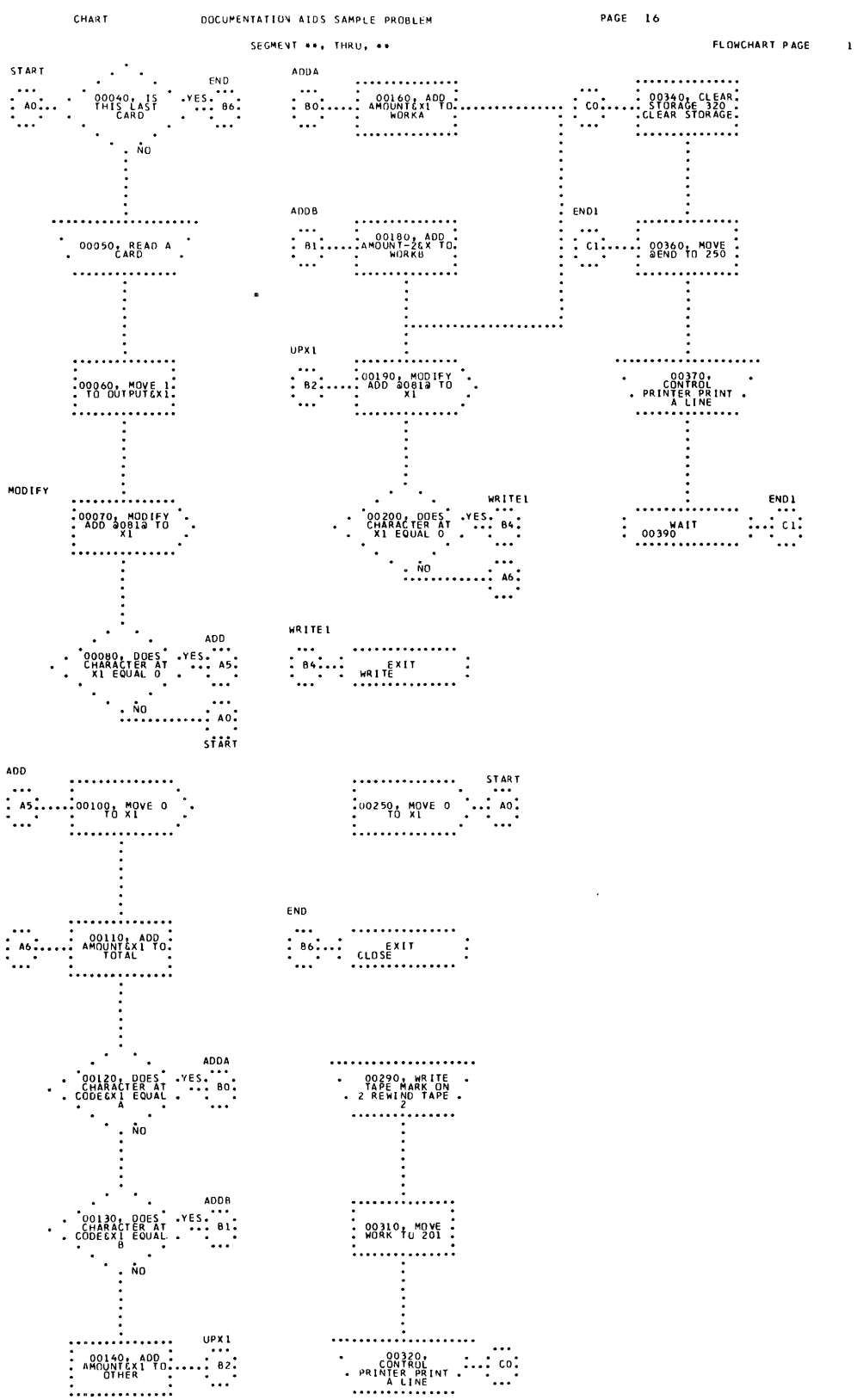


CHART DOCUMENTATION AIDS SAMPLE PROBLEM PAGE 16
SEGMENT **, THRU, **
FLOWCHART PAGE 1
START
A0
00040, IS THIS LAST CARD
YES
END
B6
NO
00050, READ A CARD
00060, MOVE 1 TO OUTPUT&X1
MODIFY
00070, MODIFY ADD @0010 TO X1
00080, DOES CHARACTER AT X1 EQUAL 0
YES
ADD
A5
NO
A0
START
ADD
A5
00100, MOVE 0 TO X1
00110, ADD AMOUNT&X1 TO TOTAL
A6
00120, DOES CHARACTER AT CODE&X1 EQUAL A
YES
ADDA
B0
NO
00130, DOES CHARACTER AT CODE&X1 EQUAL B
YES
ADDB
B1
NO
00140, ADD AMOUNT&X1 TO OTHER
UPX1
B2
ADDA
B0
00160, ADD AMOUNT&X1 TO WORKA
ADDB
B1
00180, ADD AMOUNT-2&X TO WORKB
UPX1
B2
00190, MODIFY ADD @0010 TO X1
00200, DOES CHARACTER AT X1 EQUAL 0
YES
WRITE1
B4
NO
A6
WRITE1
B4
EXIT WRITE
END
B6
EXIT CLOSE
START
A0
00250, MOVE 0 TO X1
00290, WRITE TAPE MARK ON 2 REWIND TAPE 2
00310, MOVE WORK TO 201
00320, CONTROL PRINTER PRINT A LINE
C0
C0
00340, CLEAR STORAGE 320 CLEAR STORAGE
END1
C1
00360, MOVE @END TO 250
00370, CONTROL PRINTER PRINT A LINE
WAIT 00390
END1
C1

PROGRAMMER'S INFORMATION

UPDATE PROGRAM

Abstract

The Update Program is designed to perform file maintenance on card image tapes, and is used to add or delete source statements in a program being processed by the DA System. It is also used to update the Symbolic Flowchart Language, thus providing this new language with machine maintenance capability, including updated source decks and listings.

The user, through control card options, may request a new updated source deck and/or listing. On option, the Update Program generates ascending sequence numbers in the sequence field, starting with 00010 in increments of 10. Whenever resequencing is requested, a listing showing both old and new sequence numbers (with errors flagged) is produced.

The program checks for valid sequencing, ascending 1401 collating sequence, and generates standard input files for other DA System programs.

Description

The Update program is divided into two passes.

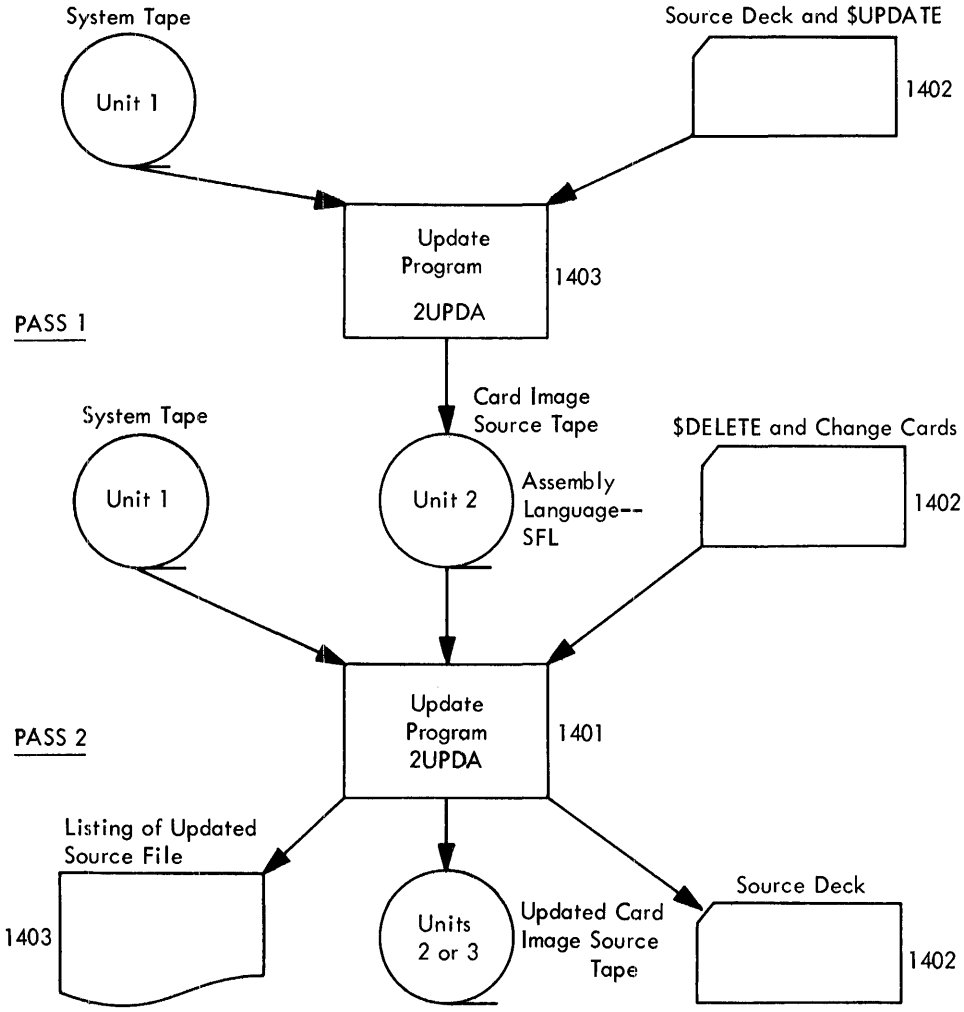
Pass 1 (2UPDA)

If the source input program is on cards, pass 1 performs a card-to-tape operation to prepare an input tape for pass 2. When a $UPDATE control card is present, pass 1 performs additional operations: it interprets the $UPDATE operands and sets the corresponding program switches. All $DELETE and change cards are put into a file to be processed in pass 2. If more than 50 change cards are submitted, the change card file is moved to tape unit 4. Sequence checking is performed on all files processed in pass 1; sequence errors are flagged on the printer.

Pass 2 (2UPDA)

Pass 2 (2UPDA) is performed when a $UPDATE, $ANALYZE, or $CHART control card is present. All optional output is generated during this pass. If SEQUENCE is indicated in the $UPDATE card, the input file is resequenced and a listing of the file is printed. If LIST is indicated, only the listing operation is performed. If DECK is indicated, the file will be punched into cards simultaneously with any other optional operations including file maintenance. At the conclusion of pass 2 all tapes are rewound, input tapes are unloaded, and control is passed to the next DA System program.

System Flow



Input/Output Description

Input

1. Card reader input file--always contains DA System control cards. Additionally, it may contain the change cards as well as assembly language program statements. The card image formats contained in this file are standard and are retained throughout the program.

2. Source program tape file - unit 2--contains assembly language or SFL statements in card image form.

Output

1. Updated source language tape file - unit 3--contains assembly language or SFL statements in card-image form. When maintenance or resequencing is performed, it contains the updated source language statements. This file may be used as input to other DA System programs, Analysis or Flowchart.

2. Updated source program card file--produced on the 1402 as a result of the DECK option in the $UPDATE control card. When maintenance or resequencing is performed, it contains the updated source language statements.

3. Update list file--produced on the 1403 and consisting of:

   a. A listing of the updated source program whose format may be any or all of the following:

      (1) Original source statements

      (2) Original source statements with new sequence numbers when the SEQUENCE option is requested (the new resequenced number is printed in the original sequence field and the old sequence number is printed to the far right of the statement)

      (3) Original source statements with out-of-sequence conditions flagged

   b. A list of all out-of-sequence conditions

   c. A list of all changes to the input file

   d. Operator control messages

   e. Diagnostic messages

ANALYSIS PROGRAM

Abstract

The Analysis Program is designed to scan an assembly language source program to provide a detailed analysis of each instruction. This analysis is produced in the following forms:

1. A flagged listing denoting instruction type

2. A cross-reference dictionary of labels and references to them

3. An analysis of operation code usage

The Analysis Program also prepares a coded assembly language tape for input to the Flowcharter.

Description - Phase I

In this phase, the second record in each pass handles S/360 input.

Pass 1 (3ANAA/3ANAB)

The input to pass 1 consists of DA System control cards and a card image tape from the Update program. The input unit is determined by a switch set in the Update program.

The current control card is examined. If it is $ANALYZE, the operands are scanned and switches are set to indicate which optional reports are requested. An error message is printed in the event of illegal options on the control card and control is transferred to the Controller (1CONA).

Machine and language combinations which have similar format characteristics are grouped together in sets for processing by the Analysis program. The sets and their components are listed below:

Set	Machine/Language
A	1401/1460 Autocoder 1440 Autocoder 1410/7010 Autocoder 7070/72/74 Autocoder
B	705/7080 Autocoder
C	1620 SPS
D	7040/44 MAP 7090/94 MAP 7090/94 FAP S/360 BAL/FAL
E	1401 SPS

The machine and language are determined and control of the program transfers to the routine that handles the indicated set. The mnemonic operation code dictionary for the particular machine language combination is read in from the system tape.

Each routine performs the following general functions: The starting location of the operand is moved to the DA record. Input records are read from tape unit 2 or 3. If the record is a comments or continuation card, the Analysis code is set to T (Transparent). The DA formatted records are written on tape unit 4.

Comments cards for each set are determined as follows:

Set	
A	Asterisk in card column 6
B	Blank cc 6--23 or a C in cc 74
C	Asterisk in cc 6
D	Asterisk in cc 1 S/360 ICTL statement
E	Asterisk in cc 8

SOURCE CARD FORMAT FOR SETS A, B, C, D, E

SET A

Page and Line	Label	Operation Code	Operands	Comments	ID
1-5	6-15 Leading blanks permitted on 1410	16-20	21-72	Two spaces from operand	76-80

SET B

Sequence Number	Name	Operation Code	No.	Operands	Comments	Flag	ID
1-5	6-15 Leading blks. permitted	16-20	21-22	23-39	40-73	74	75-80

SET C

Page and Line	Label	Operation Code	Operands	Comments	ID
1-5	6-11	12-15	16-75	Starts with 4th comma	76-80

SET D

Name Label	Blank	Operation	Blank	Operands Address, Tag, Decrement/Count	Comments	ID
1-6 Leadg. blanks perm.	7	8-14	15	16-71 Operand may begin in cc 12-16, but not past cc 16.	Starts one blank after the operand	73-80

SET E

Page and Line	Count	Label	Operation	A Operand				B Operand					Comments	ID
				Address	±	Char. Adj.	Ind.	Address	±	Char. Adj.	Ind.	d		
1-5	6-7	8-13	14-16	17-22	23	24-26	27	28-33	34	35-37	38	39	40-55	76-80

If the record is not a comments card, the label field and the operation code field are moved left-justified to the DA record. A table lookup is performed on the operation code. If the operation code is found, the corresponding Analysis code and the entry location are moved to the DA record. The expanded records are written on tape unit 4.

In addition to the above general functions, each routine handles special considerations.

SET A. In 1410 and 7070 Autocoder the channel designation for the input/output operation codes is dropped before a table lookup is performed. This is because channel designations were not included in the Operation Code Dictionaries. In 1410 Autocoder the prefix N for input/output operation codes is dropped for the same reason. After the table loopup is performed, the Analysis code is tested for the special character which indicates a change to 1401 SPS. The set is changed to E and control is transferred to SET E. The operation code is tested for blanks which indicate data defining. In this case the Analysis code and dictionary entry location of the previous statement are used and no table lookup is performed.

SET B. For one-to-one instructions it is possible to specify a register in cc 22. If this column is not blank, the register number is moved right-justified to the operation code field on the DA record.

SET C. The last character of the operation code is tested for an M, which indicates an immediate instruction. These instructions are flagged with I on the DA record.

SET D. FAP/MAP. The table-lookup subroutine is initialized in this routine to handle table table entries of seven characters. The operation code is scanned to determine whether indirect addressing is present and to locate the first position of the operand field. If the operation code is not found in the Operation Code Dictionary, the last character of the operation code is dropped and another table lookup is performed. The last character is dropped since it may be a channel designation (not included in the dictionaries). The Analysis code found in the dictionary is examined to determine whether the operation code begins or ends a macro definition. If the portion of the source program being analyzed is within a macro definition, the Analysis code is changed to T (Transparent) and the dictionary location is blanked out.

System/360 BAL/FAL. Through the use of the ICTL assembler instruction, the programmer may specify Begin, End, and Continue — in columns other than those normally used. 3ANAB checks for these instructions and makes the appropriate format adjustments. Macro definitions are handled as described in FAP/MAP.

SET E. In this set the Analysis code is tested for a comma or a period. A comma indicates a move or load machine op code. Further testing is required to determine whether it is an input/output instruction. If the first character in the operand field is %, the Analysis code is changed to K for I/O. Otherwise it is changed to -.

For each set, the first time the table-lookup subroutine is entered, the appropriate Operation Code Dictionary is read into core and two groups of binary points are calculated for use in the table lookup.

The size of the dictionary is calculated from the origin address and the high address of the dictionary. The number of entries is determined by subtracting the entry size from the dictionary size until the dictionary size is zero. The absolute binary points are then calculated. These absolute binary points are multiplied by the entry size to obtain the relative address of each binary point in the dictionary.

The table-lookup subroutine performs a binary search on the operation code. The location of the entry in the dictionary is calculated by adding the absolute binary point to the location field every time the operation code being searched for compares high. If the operation code is not found after using 14 binary points, the search is discontinued and the operation is assumed to be a user-defined macro.

If the current control card was preceded by a $UPDATE card, the input tape is unloaded and a message is printed. If the COUNT option is present, phase I--pass 2 is called; otherwise phase I--pass 3 is called.

DA format record as it appears in phase I--pass 1 for input to phase I--pass 3 of Analysis.

1-5	6	7-8	17	21	22	23-32	33-38
Serial Number	1620 Immediate Flag Position	Start of Operand	Set	7090 Indirect Addressing Flag Position	7090 Program Macro Position	Label	Operation Code

39	40-42	81-160
Analysis Code	Location	Card Image

1-5	A generated serial number
6	1620 immediate flag position
7	First column of operand--two-position field
17	Set--groups of systems with similar rules
21	7090 indirect addressing flag position
22	7090 program macro flag position
23-32	A ten-position field containing the Label
33-38	Six-position operation code
39	A code assigned to each operation code indicating the nature of the operation code. It is found in the dictionary with a table lookup on the operation code.
40-42	The number of the table entry in which the operation code was found
81-160	Source card image

Pass 2 (3ANAU/3ANAQ)

When this pass is read into core, the operation code dictionary from pass 1 remains. A second table is used to hold the tally for each operation code. Each entry in this table is three positions long and has a corresponding entry in the operation code dictionary.

A record is read from tape 4. If the field which contains the dictionary entry location is nonblank and nonzero, the corresponding entry location is calculated for the tally table, and that entry is incremented by one. A total of all the tables is kept. At end of file, the tally table is scanned and the nonblank tallies are printed with the corresponding operation code dictionary entry. During this scan, tallies are kept on the type of operation code, as determined from the Analysis codes in the operation code dictionary entries.

When the scan is completed, these totals are divided by the total number of operation codes to give the percentage of each type. The types, the number found, and the percentage are printed. The total number of operation codes is printed and phase I--pass 3 is called.

Pass 3 (3ANAV, 3ANA9/3ANAR)

At the beginning of this pass, tapes 2 and 4 are rewound.

A DA format record is read from tape 4. The Analysis code is examined to determine whether the operand of the card image is to be scanned.

If the Analysis code is a T, the statement (such as a comments card) is considered to be transparent. The sort field is blanked out and the record is written onto tape 2.

If the code is an 11 or 12 zone, the operand is scanned. The following functions are performed in the scan:

1. Input for the Flowchart program is created. This consists of placing the first three operands in the card image into special fields on the record. Also added to the record is a count of the number of operands present and a flag which indicates the nature of the first operand.

   The flags are:

   S Simple symbol
   R Simple relative addressing
   L Location counter
   D D-modifier (1401 SPS only)
   O Other

   If the Analysis code contains a 12 zone, the first operand is placed into the first field as follows:

   SYMBOL ±m

where SYMBOL is either a simple symbol or a location counter notation (*), and m is a numeric adjustment. The symbol is left-justified and the adjustment is right-justified unless the operand is flagged O. In this case or if the Analysis code contains an 11 zone, the first 10 characters of the operand are moved into the first field, as is always done with fields 2 and 3.

2. If the current control card is $ANALYZE, the scan also determines the nature of the statement, whether indexing, relative addressing, indirect addressing, or complex operands are being used. Flags indicating the presence of any of these functions are placed in temporary storage locations in the record.

   Whenever the dollar sign symbol ($) is found within the first operand, a switch is set. If the first character of the first operand is a $, it is dropped and the remainder of the symbol is left-justified in field 1.

3. If any reference options (cross or operand) are present on the $ANALYZE card, a table is created containing the symbols which are present in the operand field. The entries are variable length and are separated by record marks. A count of the number of entries is made.

   When a $ is found to be present within a symbol, a count of the number of characters which precede the $ is placed within a field, the first position of which corresponds to the first symbol, the second to the second symbol, etc.

Scan Routine -- Special Considerations

The following special considerations are made in the scan routine for particular languages:

1410 Autocoder. Whenever a $ is found in the first operand, the suffixing character, which may be blank, is moved to the 10 position of the field, the $ is dropped, and the remainder of the symbol is left-justified. Thus, the operand A$SYMBOL would appear in Field 1 as SYMBOL___A. This facilitates cross-referencing in the Flowchart Program.

7040/7044 - 7090/7094 FAP/MAP. When a symbol contains a $ preceded by more than three qualifying characters, the excess characters are dropped from the qualifier in the symbol table. Because of field size limitations, a symbol may not be more than ten characters. Thus, the symbol ABCDEF$SYMBOL would appear in the symbol table as ABC$SYMBOL.

1401 SPS. Although the SPS source card is of fixed format, the operands are assigned to the fields as though it were of variable format. Thus, if there were a symbol in the A field, a blank B field, and a D-modifier present, the D-modifier would appear in the second field — not the third. When the D-modifier is in the first field, a special flag, D, is assigned to the first operand flag.

705/7080 Autocoder. Prefixes, such as the prefix for indirect addressing (I), are not treated as the first operand. The first operand is the one which follows and is the one moved to field 1.

System/360 Assembly Language. A symbol with hexadecimal adjustment will be flagged as relative and complex.

Each set has a corresponding set of rules used by the Analysis program to scan the operand field. Rules for each set are outlined below.

Set	Symbol	Operators	Operand Separators
A	1. First character alphabetic 2. No special characters 3. No blanks 4. Limit 10 characters	+ or -	, or blank
B	1. Alphabetic, numeric, blanks 2. No special characters 3. L, H., R, S, I, modifiers 4. Limit 10 characters 5. @actual address	+, -, *, /	,blank or)
C	1. At least one character must be nonnumeric 2. Special characters permitted. = @. / 3. Limit 6 characters	+, -, *, /	, or blank
D	1. Alphabetic and numeric 2. Special characters permitted ( ) 3. No blanks 4. Limit 6 characters	+, -, *, /	, blank or (
E	1. First character alphabetic 2. No special characters 3. No blanks 4. Limit 6 characters	+ or -	Fixed position
S/360	1. First character alphabetic 2. No special characters 3. No blanks 4. Limit 8 characters	+, -, *, /	, or blank

After the operand scan the analysis code is again examined. If it is $, the program transfers to a routine which determines whether qualification* is beginning or ending. If it is *, the program leaves the qualification mode.

*Qualification is that function performed by some of the assemblers to uniquely define the labels and operands of a section of a program by either prefixing or suffixing a character or symbol to each of them. This function is initialized by the assembler instruction SFX in 1401/1460 Autocoder and 1410/7010 Autocoder, HEAD in 1620 SPS, QUAL in 7040/7044 MAP and 7090/7094 MAP, and HEAD or HED in 7090/7094 FAP. Qualification is terminated by the same assembler instruction with a blank operand or, in the case of MAP, by the instruction ENDQ.

When the program is in the qualification mode, the label field and the first operand field are qualified. For set A languages, this is accomplished by placing the suffixing character in the last position of the label. For sets C and D, the prefixing character or symbol, followed by a $, is prefixed to the symbol. In FAP or 1620 SPS prefixing is not done if the symbol is six characters long. The first operand field is not qualified when a $ has been found to be part of the symbol.

The flagged listing is printed if the current control card is $ANALYZE without the CROSS or OPERAND options. Unless these options are present, the sequence or page-line number is moved from the card image to the sequence field in the DA records. The card image is dropped from the records. These input records to the Flowchart Program are written on tape 2.

If the language is 705/7080 Autocoder, any blanks within the symbol in the sort field, the label field, or the first operand field are replaced with the special character colon (:). This character, which does not print on the 1403, is needed because blanks terminate the scan of symbols in the Flowchart Program.

When the OPERAND or CROSS options are present, a file is created which is sorted in order to produce the reference reports. Each record contains a sort field, the rightmost position of which is called the sort code.

There is one record produced for every input source statement. This is called the source record and has a sort code of 0. An additional record is created for each symbol in the symbol table. This is called a reference record. The sort code for these records corresponds to the position of the symbol within the table. The symbols in the symbol table are qualified if the program is in the qualification mode.

If the current control card is $CHART, the Flowchart program is read in. If the current control card is $ANALYZE with no reference options, the next control card is read and control is transferred accordingly.

If the current control card is $ANALYZE with reference options, phase II-- pass 1 of Analysis is called.

Description--Phase II

Pass 1 (3NALA, 3ANLB)

A three-tape sort is performed on the records created in phase I.

Pass 2 (3ANLC)

The DA format records input to this pass are of two two types: source records (which represent a source statement) and reference records (which represent an operand reference to a labeled source statement). The source records are distinguished by the fact that they have a zero sort code. During this pass, two records are maintained in core--one in an input and the other in an output area. The first records on the incoming file will contain a blank sort field (not considering the sort code as part of the sort field). These records represent those source statements which were unlabeled.

The output file created in this pass has sequence numbers in the sort field. For each incoming reference record, the card image of the last source record is written out with the reference record's own sequence number and sort code in the sort field.

Records are read from tape unit 2 into the input area. Source records are moved to the output area. When the symbol in the sort field of the input record is nonblank and the CROSS option is present in the $ANALYZE card, a line of the cross-reference dictionary is printed. Otherwise, the serial number in the input area is moved to the sort field of the output area and the output record is written into tape unit 3.

If the input record is a reference record, the symbol in the sort field is compared with the label field in the output area. If they are not equal, the symbol in the input area is a virtual symbol--that is, no corresponding label was found in the source program being analyzed, in which event a new record is read in. When the symbol in the input area compares equal with the label in the output area, the sort code in the input area is moved to the sort code field in the output area. If the CROSS option is present, a reference line is printed on the cross-reference report. If the OPERAND option is present, the serial number in the input area is moved to the sort field in the output area, and the output record is written into tape unit 3.

Pass 3 (3ANLD, 3ANLE)

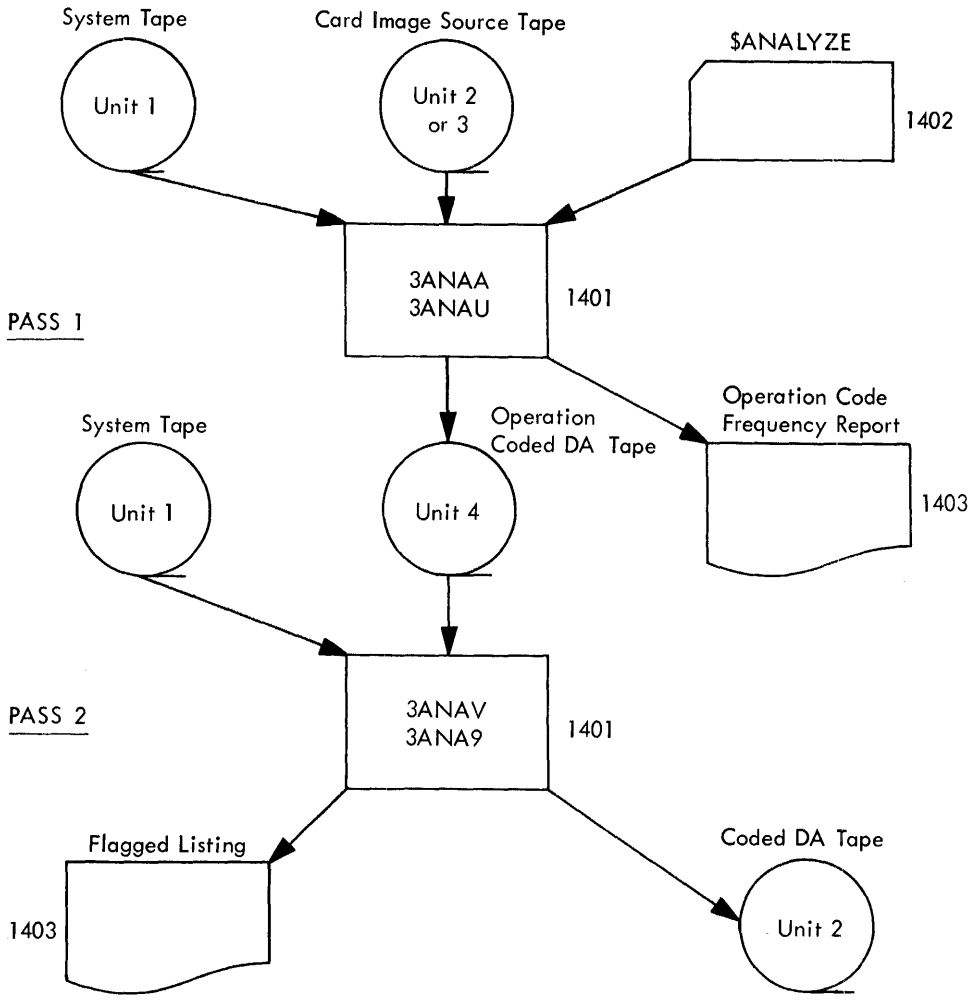
A three-tape sort is performed on the sort field of the tape generated in pass 3.

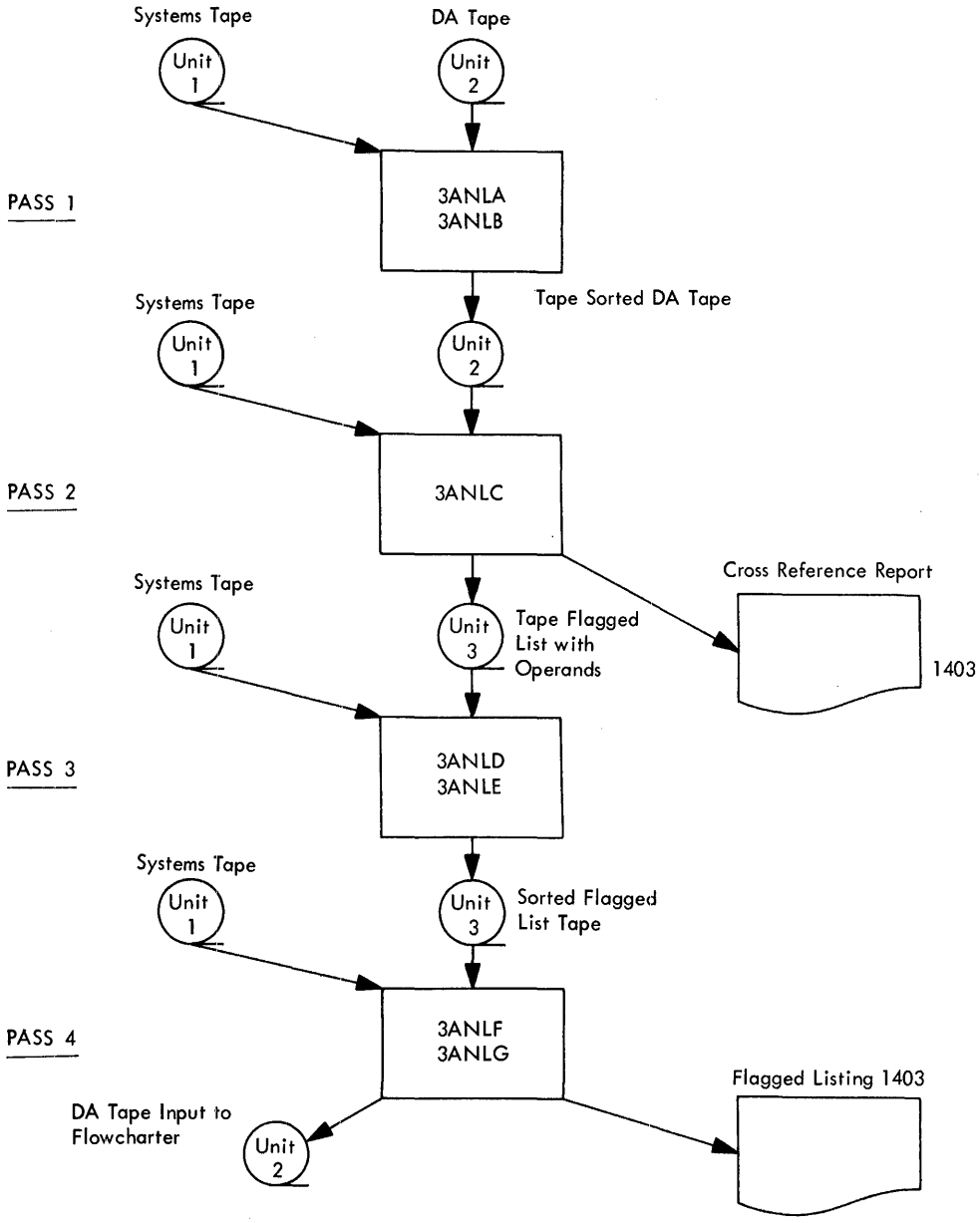
Pass 4 (3ANLF, 3ANLG)

At the beginning of this pass tapes 2 and 3 are rewound. A record is read from unit 3. If the sort code is zero, a line of the flagged listing is printed. The original sequence number in the card image is moved to the sequence field unless the record is transparent. For the set B languages, any embedded blanks in the label field, field 1, or the sequence field are replaced with the special character colon. The card image is truncated from the DA format record, which is then written into unit 2.

If the sort code is nonzero, a reference line is printed on the flagged listing with operands report and a new record is read in. At end of file, if the current control card is not $ANALYZE, a new control card is read and control is transferred accordingly.

System Flow





Analysis--Phase II

Input/Output Description

Input

1. Card reader input file--contains DA System control cards.

2. Source program tape file--unit 2 or 3--contains assembly language statements in card image form as produced by the Update program.

Output

1. DA System format tape--unit 3--the coded representation of the source statements used as input to the Flowchart program. The format is an 80-character record.

1 - 5	Serial number
6 - 17	Working area
18 - 22	Sequence number
23 - 32	Label
33 - 38	Mnemonic operation code
39	Analysis code
40 - 42	Position of operation code in dictionary
43 - 44	Operand codes
45 - 54	Operand 1
55	Plus or minus
56 - 60	Displacement of operand 1
61 - 70	Operand 2
71 - 80	Operand 3

2. Analysis reports file--produced on the 1403 and consisting of:

   Cross-reference report

   Flagged listing

   Operation Frequency report

   Operator control and error messages

   Diagnostic messages

   The format of the first three is shown by the sample problem.

FLOWCHART PROGRAM

Abstract

The Flowchart Program is designed to generate a flowchart of an existing source program. The flowchart produced represents the gross logic of the source program and, therefore, can be used as a guide for reprogramming in a higher-level language (for example, COBOL or FORTRAN).

The Flowchart Program scans assembly language statements which have been coded by the Analysis Program and generates a language called Symbolic Flowchart Language (SFL).

SFL is then processed producing a detailed flowchart of the original program.

Description--Phase I

Pass 1 (4CHRA)

The input to pass 1 consists of DA System control cards and a tape which is either SFL or a DA format tape from the Analysis Program. If the input is an SFL tape, control is passed to phase II immediately after processing the $CHART card. If the input is from Analysis, pass 1 performs the following processing:

1. $SEGMENT cards (if present) are scanned to determine the segmentation to be performed.

2. Each operation which was looked up in a table by Analysis is found in a corresponding operation table in pass 1 of Flowchart. The pass 1 operation table entries contain a six-character code, which is entered into the DA record and controls the processing in the remaining passes of phase I.

3. The length of each statement is computed and entered into the DA record; an area number is generated for each statement and placed in the DA record. All instructions within a known length area are assigned the same area number.

4. Nonprocedural instructions, such as data defining and assembler control, are deleted from the DA format tape.

Pass 2 (4CHRS)

The input to pass 2 is the DA format tape from pass 1. Pass 2 generates label dictionaries used in passes 3 and 4 to reduce simple relative addresses of branch instructions.

The label dictionary is generated for each segment by entering the labels of all procedural statements. Each entry in the label dictionary is 20 characters in length and consists of (1) the label, (2) the area number for the instruction, (3) a label type code, and (4) the forward and backward displacements in location counter units to the closest label or change in area number.

If the input program is segmented, the label dictionaries are written on a work tape (unit 2). When the program is not segmented, the label dictionary remains in core for subsequent processing. After processing the last segment, control passes to pass 3.

Pass 3 (4CHRT)

The input to pass 3 is the DA format tape from pass 1 and the label dictionary from pass 2 for each segment.

Pass 3 processes all branch instructions with simple relative addresses. A simple relative address has either of the two following forms:

Label ± n

or

*± n

where "Label" is the symbolic address of some instruction, n is some numeric constant, and * is the value of the location counter for this instruction.

If the relative address refers to the same location as a label already in the label dictionary, no label generation by pass 3 occurs. Otherwise, pass 3 normally generates a label for the instruction to which the relative address refers and inserts the generated label in the dictionary at the correct position.

Pass 3, in addition, sets the type code for labels referred to by a subroutine call.

If the DA format input consists of only one segment, the expanded dictionary remains in core for pass 4 processing. Otherwise, the expanded label dictionaries are written onto tape (unit 3).

Pass 4 (4CHRU)

The input to pass 4 is the DA format records from pass 1 and the expanded label dictionary from pass 3. Pass 4 completes the derelativization process by entering into the DA record all labels generated in pass 3.

All simple relative addresses of branch instructions are processed in pass 4.

If the relative address refers to a label in the label dictionary, that label replaces the relative address in the branch instruction. Otherwise, if the relative address does not refer to any label in the label dictionary, the type of instruction is changed to a flowchart EXIT type.

In addition, each branch to an instruction classified as a subroutine ENTER is changed to a SUBRT type. Each instruction to which a SUBRT call occurs has its type changed to an ENTER.

Output at conclusion of pass 4 are the derelativized DA format records.

Pass 5 (4CHRV)

The input to pass 5 is the derelativized record in DA format. Pass 5 uses the information in each DA record to generate the SFL card images. Information associated with each operation is used to determine both the type of SFL operation and the comment to be generated.

Comments are generated by constructing the comment from a string of characters called a comment skeleton. Comment skeletons designate which information is to be used from the DA record in forming the comment, and also specify the additional words, such as READ TAPE and COMPUTE, which are to be generated as part of the comment.

Each segment produces an SFL program bounded by an SFL JOB and END card. The SFL tape (unit 3) is then rewound and serves as input to the second phase.

Description--Phase II

Pass 1 (4CHTB)

This pass reads a Symbolic Flowchart Language program bounded by a JOB and END statement. All commentary statements are flagged to avoid further processing, and are not required again until the flowchart is generated in pass 8. The chart mode statements are processed, expanded, and written on a work tape. Logical connector operations (YES, NO, and GOTO) are combined with the box statement from which they exit. Final page and matrix positions are established for each chart box. Comment information in chart mode instructions is analyzed for errors and arranged in a format for final printing. The source program is also analyzed for logic errors such as multiple GOTO exits. The source program is printed and/or punched, as specified by user options.

Pass 2 (4CHTC)

This pass reads the chart mode statements and constructs a table of labels with the page and matrix positions at which each label is defined. During passes 2, 3, and 4, this label table uses the major portion of core storage. If the label table overflows, an error message is printed and the segment is bypassed. The chart mode statements are not altered in this pass.

Pass 3 (4CHTD)

Pass 3 reads the chart data and examines each statement to determine whether a logical connection is being made to another symbol by the symbolic operand of a GOTO, YES, or NO operation. If so, the label table is searched and, if the label is found, the page and matrix position of the label are inserted into the chart mode record. If the label is not found, an error message is printed. If a logical connector refers to a label on a different page, the label is flagged in the label dictionary as a label requiring the generation of an off-page entrance arrow. The chart records are written onto a work tape.

Pass 4 (4CHTE)

Pass 4 reads the chart mode records into core and examines each record for a label definition. When a label is encountered in the record, the corresponding label table entry is examined to determine whether an off-page entrance pointed arrow is required for this label. A flag is then set in the record establishing whether a round or pointed arrow entrance is to be generated for the record. The modified chart records are written onto a work tape.

Pass 5 (4CHTF, 4CHTG)

This pass sorts and prints the label table, noting the page and matrix position at which each label is defined. Labels which are defined more than once or have not been referenced by a connector operation are flagged. All references to a label are also printed by passing over the chart mode records and saving the page and matrix position of each reference. This is the last pass in which the label table is used.

Pass 6 (4CHTH)

This pass reads the chart records into core, rearranges 30 records at a time into row by column order, and blocks the output records by three, corresponding to a chart row. These records and the commentary statements are written onto a work tape.

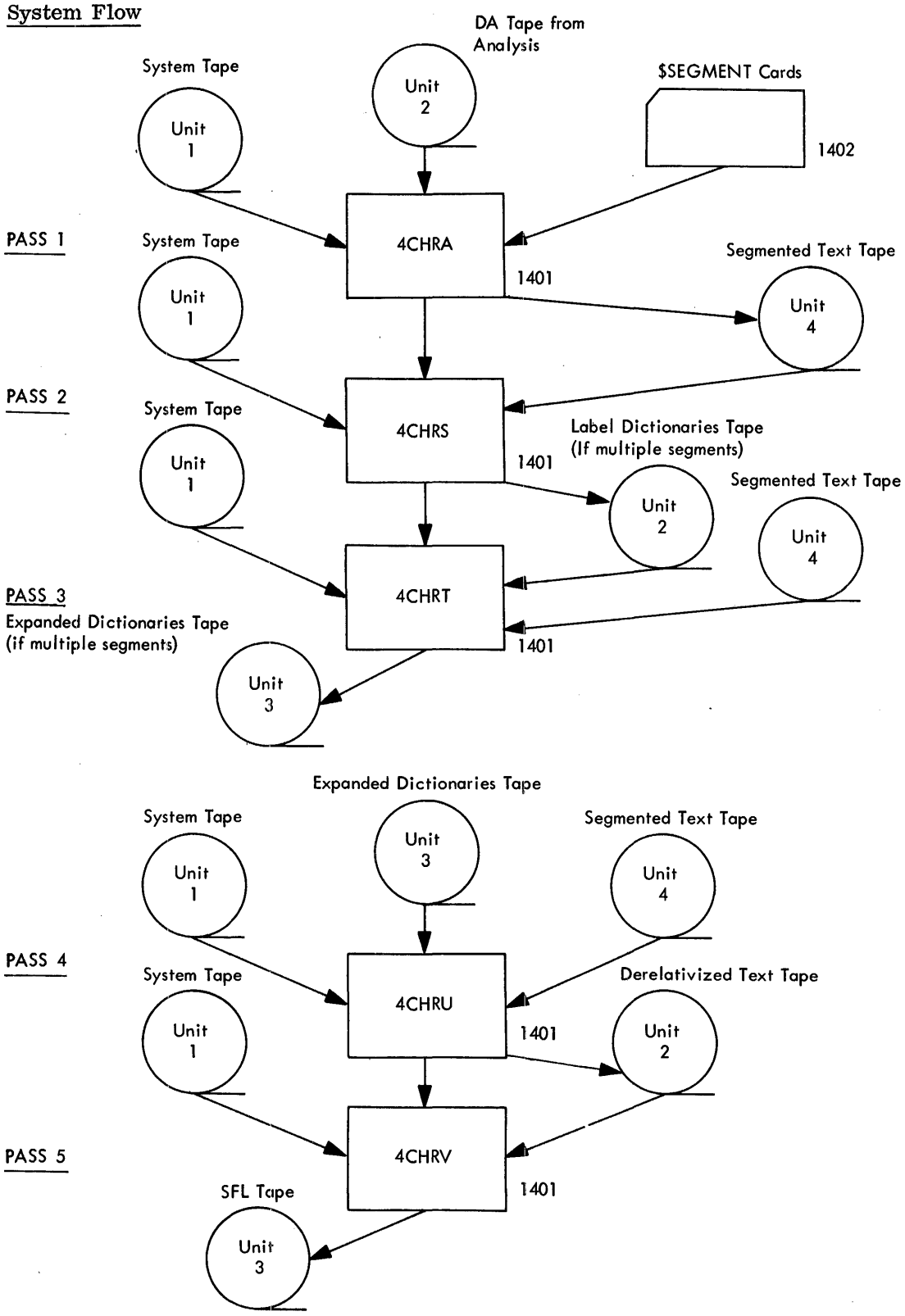
Pass 7 (4CHTI)

Pass 7 reads the chart data into core one page at a time (30 boxes or ten physical records) and constructs an internal matrix table. Several passes are made on the matrix table which analyze possible connector paths between boxes. A line table is constructed which contains flags representing different segments of each printed line. The flags contain all information required to generate the skeleton portion of the chart page--that is, the boxes, lines, and arrows. The line table information for each page is written onto a work tape. At this point, one of the work tapes contains line information pertaining to the flowchart and another contains comment information pertaining to the flowchart.

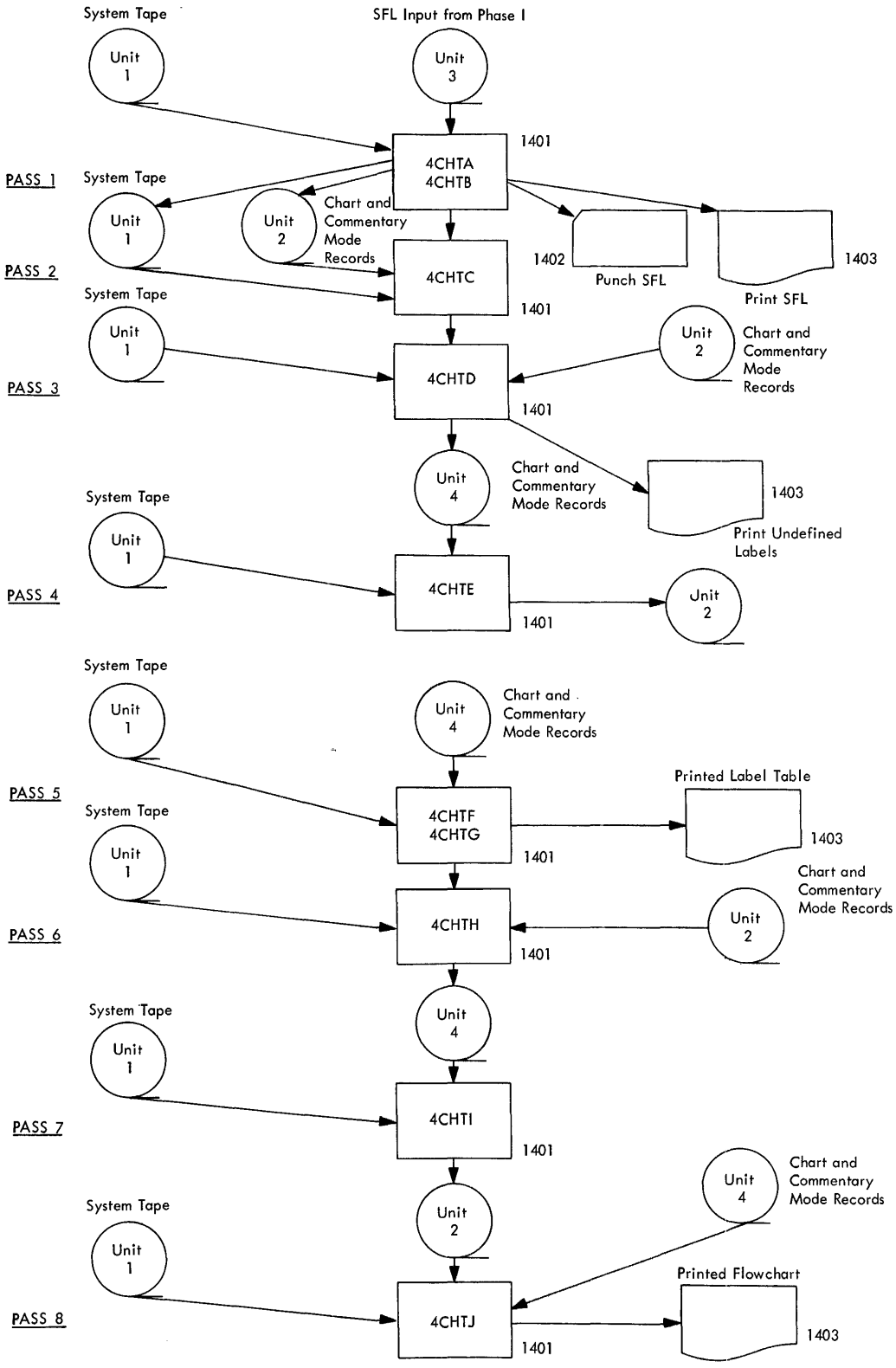
Pass 8 (4CHTJ)

This pass reads the line and comment records to generate a flowchart. For chart mode pages a print line is formed from the line table information and the comments are inserted into each box. The labels and matrix positions are also inserted into the print lines as required.

System Flow



Flowcharter--Phase I



Flowcharter--Phase II

Input/Output Description

Input

1. DA System format tapes. The format of the records of phase I of Flowchart is the DA System format described in the Analysis. The working area is used by the Flowchart as follows:

Column	Usage
6 - 8	Not used
9	1401/1410 chaining condition
10	Pass 1 processing code
11	Pass 1 length code
12 - 14	Instruction length
15 - 17	Area number assigned to instruction

2. SFL input file--unit 3. This tape format is generated in phase I of Flowchart to be used as input to phase II. The 80-character record format is:

Column	Usage
1 - 5	Sequence number
6 - 15	Label
16 - 20	Operation field
21 - 72	Operation field
73 - 80	Ignored

3. Card reader file. This file contains DA System control cards.

Output

1. Card punch output file--generated by an option. This is the Symbolic Flowchart Language program generated by phase I. The SFL format is described above.

2. Symbolic Flowchart Language tape file--unit 3--always generated when the input is an assembly language program. The contents of this file are the same card images as the card punch output file.

3. Flowchart file--produced on the 1403 and consisting of:

   a. The flowchart

   b. A cross-reference dictionary of flowchart labels giving their page and chart locations

   c. A printout of the Symbolic Flowchart Language program

   d. Operator control and error messages

   e. Diagnostic messages

The record formats for chart and commentary mode records created internally in Flowchart phase II are as follows:

Chart Mode Record

In pass 6 these are blocked three to a physical record.

	0	Internal Seq. No.	Chart Page	Coordinate	Op Code	Statement Label	BCD Op Code	Symbol Text
Position	0	1-5	6-9	10-11	12	13-22	23-27	28-92

	Exit 1 Label	Page No.	Exit 1 Coord.	No or Yes	Exit 2 Label	Exit 2 Page No.	Exit 2 Coord.	Exit 2 No or Yes
Position	93-101	103-106	107-108	109-111	112-121	122-125	126-127	128-130

	Connector Flags
Position	131-137

For chart mode records position 12 has the following meaning:

OP CODE	FUNCTION
E	EJECT
J	JOB
N	SKIP
S	SPACE
Z	END
0	NOTE
1	BLOCK
2	IO
3	DECID

OP CODE	FUNCTION
4	MODFY
5	PREDF
6	TERMINAL OPERATION
7	GOTO
8	SUBRT
9	YES or NO

Commentary Mode Records

	0	Internal Seq. No.	Op Code	# of lines in space operation	Page No.	Commentary Information	Not Used
Position	0	1-5	6	7-8	9-12	13-69	70-137

For commentary mode records position 6 is either an * or S to indicate comments or spacing, respectively.

The record format for line records is as follows:

Line Type Record

		Page No.	Coor-dinate		Flags for Line Gen-eration	Page No.	Coor-dinate		Flags for Line Gen-eration	Page No.	Coor-dinate		Flags for Line Gen-eration
Position	1	4	5-6	7	8-40	41-44	45-46	47	48-80	81-84	85-86	87	88-120

Each box environment consists of a 16 by 40 character print position matrix. These flags are used for line generation within each box environment. The line information records are blocked three to a record. Each physical record contains information for a complete row of the flowchart.

Additional Flowchart Options

During checkout of the DA System it was desirable to implement four additional $CHART card parameters which may prove valuable to a system user. These parameters may be specified in any order along with DECK and LIST options.

Parameter	Function
NOCHART	To bypass the generation of the flowchart. The listing and/or deck of the SFL Program may be obtained, but no cross-reference list or flowchart is produced.
NOSTOP	To suppress the stopping which occurs at the beginning and ending of a run to mount the special paper carriage tape.
NOCROSS	To suppress the printing of the flowchart cross-reference list.
NOSAVE	To prevent the normal rewind-unload operation on the SFL tape unit 3. When this option is used, the SFL tape is considered a work tape and only rewound.

VERIFY PROGRAM

Abstract

The Verify Program is designed to help the programmer determine that the source deck is in agreement with the current object deck. It processes an object program deck generated by the following assembly languages:

1401 SPS
1401/1440/1460 Autocoder
1410/7010 Autocoder
705/7080 Autocoder
7070/7072/7074 Autocoder

Verify generates a storage map and identifies overlay patches.

Description

Pass 1 (5VERA)

The $VERIFY control card is checked for the presence of options. If the LOADER option is present, the number of cards of the smaller standard loader for the machine specified is read in, and the next card is checked to see whether it is a loader card. If it is, the number of cards to equal the larger standard loader are read in.

The lengths of the standard loaders are:

1401/1440/1460	3, 4, 5 (special testing)
1410/7010	5, 9
705/7080	5, 9
7070/7072/7074	5, 10

Clear storage and bootstrap cards are read in for the 1401 whenever present. The loader option is not needed.

Pass 1 then branches to the subprogram written for the particular machine and language combination specified. The following are separate subprograms:

1401 disk AUTO	DISK (option)
1401 tape AUTO	TAPE
1401 SPS	RSPS
1410/7010 AUTO	R1410
705 AUTO	R705
7080 AUTO	R7080
7070/7072/7074 AUTO	R7070

Each subprogram performs the same general function: It reads the object program and places the program on tape basically in the format of the assembly listing for the machine and language specified. The object program is read in and processed one card at a time. The card sequence number is picked up first, then the card is tested to see whether it is a special type--executive, transfer control (705/7080), or end card. If not, the high-order location of the program data on the card is picked up, decoded if necessary, and stored in the record area. The data itself is then processed and set in the tape record area, one instruction at a time, and the record is written on tape.

If the data can be determined as other than an instruction, it is picked up in specified segments (12 for 1401 through 1410, 5 for 705 and 7080). The method for picking up the data is different in each subprogram, because of the format differences of the object programs. The 1401 subprograms share a common data processing loop, which is set up as a separate subprogram. The three 1401 subprograms use the same subprogram to decode locations called DECODE, and the R705 and R7080 subprograms both use a subprogram EXPAND to determine the location and ASU for each instruction.

Special procedures implemented in the individual subprograms are described as follows:

All 1401 and 1410. Groupmark/wordmarks are denoted as GMWM's. A groupmark which is found as a d-modifier is printed as GM. When an execute or end card is read, one record is written with the execute instruction and no locations.

TAPE and RSPS. If no sequence numbers are present on the deck, sequence numbers are generated by VERIFY for the tape records. If any cards are present between an execute card and its bootstrap card, they are printed out at the time they are read with a message identifying the execute card by its sequence number.

R705 and R7080. Expanded card format is accepted and processed in the same format as condensed cards. Constant data format differs from instruction data. The number of characters in the record is placed in the ASU position.

R7070, R705 and R7080. When a transfer control, execute, or end card is read, each instruction on the card is processed with locations ascending by one from a base of zero.

When a $DAEND control card is read in, signaling the end of the deck, control is returned to the main pass, which writes a tapemark on the tape with the data on it, rewinds the tape, and calls in pass 2.

Pass 2 (5VERB, 5VERC)

This pass of VERIFY is the DA Sort Program, which sorts the tape records on the location field to determine identifiable overlay patches.

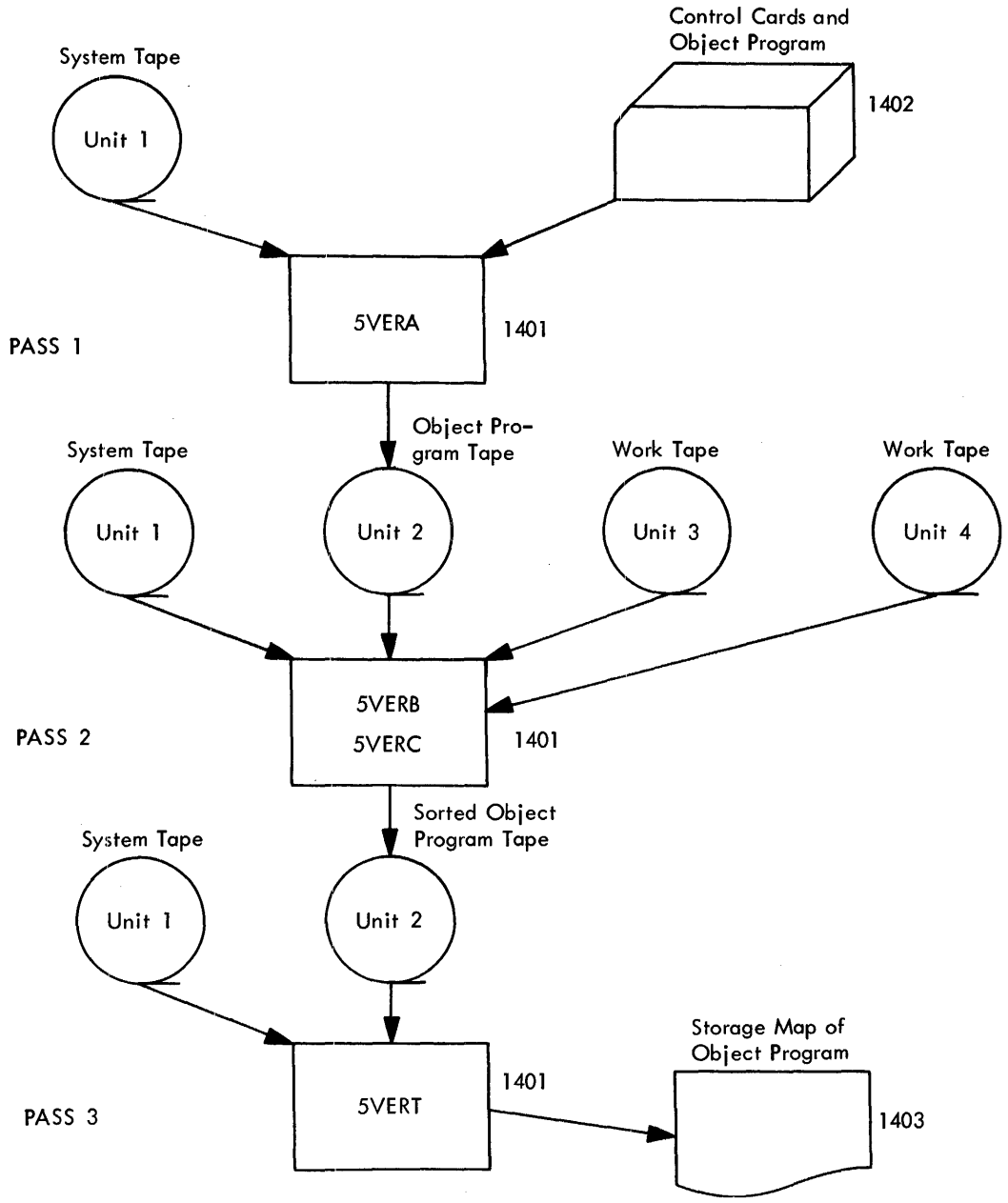
Pass 3 (5VERT)

The mnemonic-actual op code dictionary used by the machine and language specified is read into core from the DA Systems Tape. If the 1410 is specified, a special subroutine is executed which checks for the need of word separators in the table, and creates them when flagged. The heading record is read from the object program tape and set up to be printed at the top of each page. The binary points needed for the table lookup are computed.

The program then processes the sorted object program tape. A record is read in and checked to determine whether it has been identified as constant data by pass 1. If so, it is printed as read in without a mnemonic op code. If not, processing continues. If the machine specified is one of the 1400 series, the actual op code is tested for certain mnemonics before going through the table lookup. For 1401/1440/1460, the mnemonics are BCE, BBE, and two special I/O's. For the 1410/7010 they are BCE, BBE, CC1, CC2, BEX1, and BEX2. If the actual code is one of the mnemonics tested, that mnemonic is printed with the record. Otherwise, the actual op code goes through a table search to find the mnemonic. If a mnemonic is found, it is printed with the record. If not, the mnemonic code is left blank and the record is printed. In the case of the 1401/1440/1460, the data is placed in the format of constant data.

The print subroutine checks for execute or transfer control cards and ejects to a new page after one is encountered. On each new page the heading is printed. Each data record is tested to determine whether it is an overlay patch. If it can be so determined, a flag of** is printed just left of the location. When the tapemark is sensed on the object program tape, the page is ejected, the $DAEND card is printed on the new page, and control is returned to 1CONA.

System Flow


Control Cards and
Object Program
System Tape
Unit 1
1402
5VERA
1401
PASS 1
Object Pro-
gram Tape
System Tape
Unit 1
Unit 2
Work Tape
Unit 3
Work Tape
Unit 4
5VERB
5VERC
1401
PASS 2
Sorted Object
Program Tape
System Tape
Unit 1
Unit 2
Storage Map of
Object Program
1401
5VERT
PASS 3
1403

Input/Output Description

Input

The card reader file contains DA System control cards and standard object decks generated by:

1401 SPS
1401/1440/1460 Autocoder
1410/7010 Autocoder
705/7080 Autocoder
7070/7072/7074 Autocoder

Output

The output file from the Verify program is produced on the 1403 and consists of:

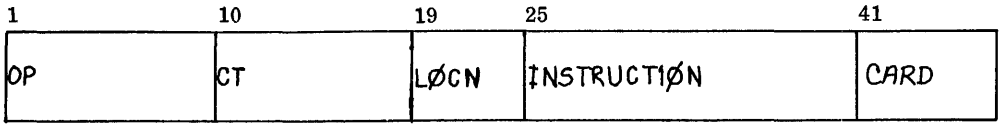
1. A storage map of the object program

2. Operator control messages

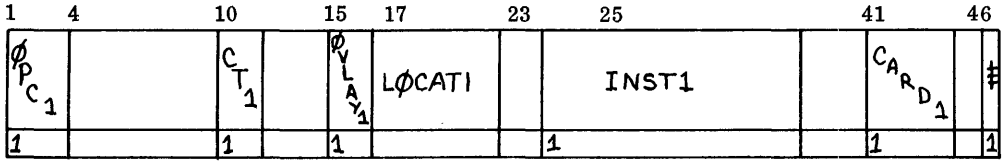
3. Diagnostic messages

Tape Record Formats

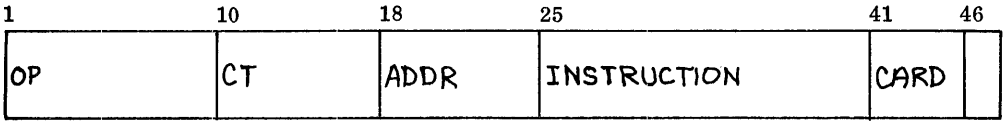
Tape record formats created in Verify pass 1 for processing in pass 3 are:



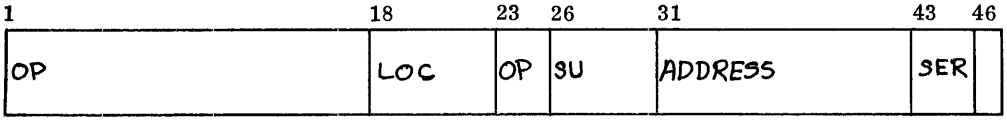
1401/1440/1460 Page Heading Record



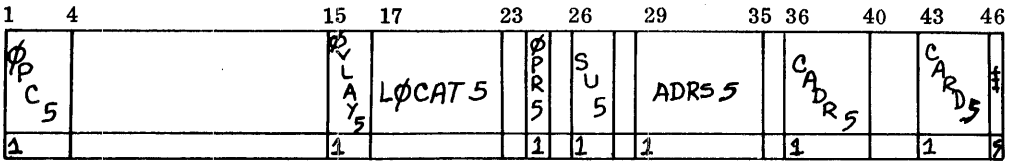
1401/1440/1460 - 1410/7010 Tape Record Format



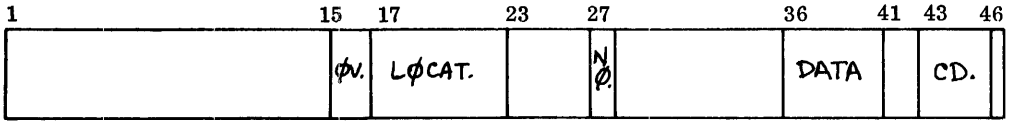
1410/7010 Page Heading Record



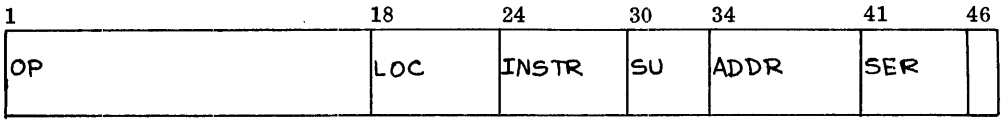
705 Page Heading Record



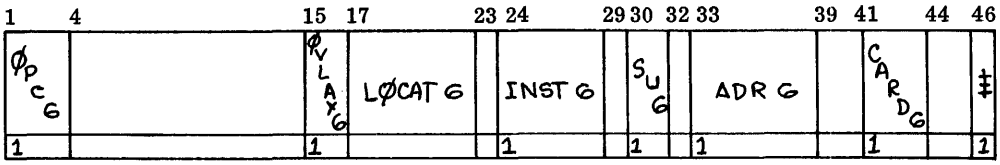
705 Instruction Tape Record Format



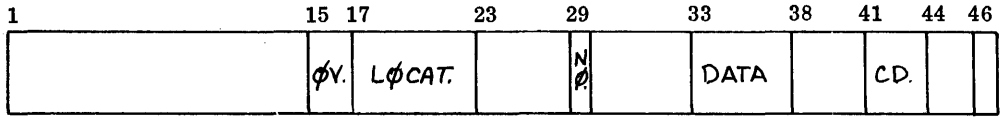
705 Constant Data Tape Record Changes



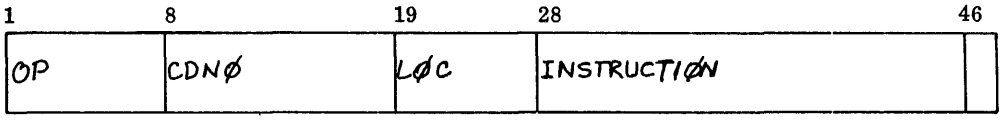
7080 Page Heading Record



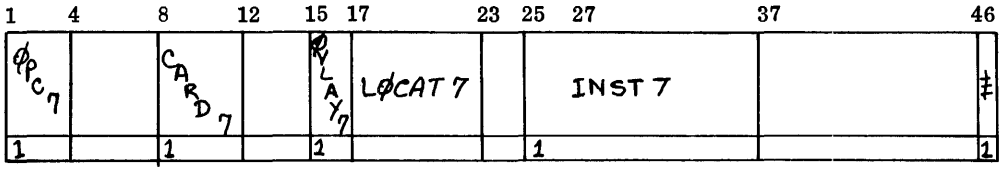
7080 Instruction Tape Record Format



7080 Constant Data Tape Record Changes



7070/7072/7074 Page Heading Record



7070/7072/7074 Tape Record Format

DOCUMENTATION AIDS CONTROLLER

Abstract

The Documentation Aids Controller consists of:

1. Resident I/O routines. These routines remain in storage during execution of all system programs. They perform all system I/O including calling in of the programs on the system tape. It is the first record on the system tape.

2. Program Selector (1CONA). This program is automatically called in by the Resident I/O routine at the beginning of a system run. It reads and analyzes the $DAJOB card and calls the next system program on the basis of this analysis. This program and its header are the second and third records on the system tape.

Resident I/O Routine Description

Tape

Entering at SYSIO, the address of the constant following the branch to SYSIO is stored. The content of the index register 3 is saved and restored at the end of the I/O. The address of the constant is moved to index register 3. The constant is moved into the I/O instruction and the I/O error counter is set to zero. If the I/O instruction is a write, the error counter is set to 7 so that trying the operation three times will cause the counter to overflow.

The I/O is performed and the address of the terminating groupmark/wordmark is stored for use in other programs. The exit is initialized to the address of the instruction following the I/O constant, index register 3 is restored, and control is returned to the user.

If a transmission error is encountered during the I/O, the tape unit is backspaced, the error counter is incremented, and, if no arithmetic overflow occurs as a result of the add, the I/O is reexecuted. If overflow does occur and the I/O is a read, the machine stops at halt 1. Pressing START causes the operation to be retried ten more times. If the operation is a write, the tape is skipped forward and the I/O reexecuted.

Printer Page Overflow

This routine may be entered directly by one of the system programs or it may be entered from the print routine as the result of printing the last line on a page.

Entering at EJECT stores the return address. The overflow switch is turned on. This switch may be tested by the system program if special spacing or multiple heading lines are required. The line counter is reset to 01, and the page counter is incremented and moved to the print area. The heading information is moved out of the punch area to the print area. The carriage is restored to channel 1 and the print routine is entered at 1NCRLC to print the heading.

Print

Entering the print routine stores the return address, turns off the overflow switch, and increments the line counter. The contents of the print area are printed and the line count is compared with the maximum. If the line count is less than maximum, control is returned to the user after the print area is cleared.

If the line count is equal to the maximum, control is given to the EJECT routine.

Read

The return address is stored, a card read instruction is issued, and control is returned to the user.

Punch

The return address is stored, a card punch instruction is issued, and control is returned to the user.

Message

This routine is used to print messages to the operator. The return address is stored, the carriage is restored to channel 1, and the program branches to PRINT. On returning from PRINT the carriage is restored to channel 1 to enable the operator to read the message, and control is returned to the user.

Program Call

All DA System programs are called by executing a B SYSLNK followed by the fifth-position identification of the desired record. The mode of I/O is changed to "Load" since the system tape is written with wordmarks. The next available record on the system tape is read (using the tape I/O routine), and identification in this tape record is compared with the constant following the branch to SYSLNK.

If the compare is equal, the requested program is the next record on tape. An unequal compare indicates the program is farther along the tape, and the tape read and compare routine is again executed. An end-of-file condition during this loop indicates a system error; the program called was not on the tape between the program call and the terminating tapemark.

When the correct header is found, the addresses necessary for loading the program are extracted. The tape I/O constant is initialized and the program is read in using the tape I/O routine. The I/O mode is changed back to "Move" and the first instruction in the new program is executed. The header information remains in the controller area.

Program Selector Description (1CONA)

The Program Selector reads the first card in the reader. If it is a $DAJOB card, the operands are extracted using a left-to-right scan technique. The program identification (third operand) is moved to the punch area, where it is used for page headings. The other two operands are compared with a table of machines and languages. When an equal compare is made, the proper code is put into the machine or language switch.

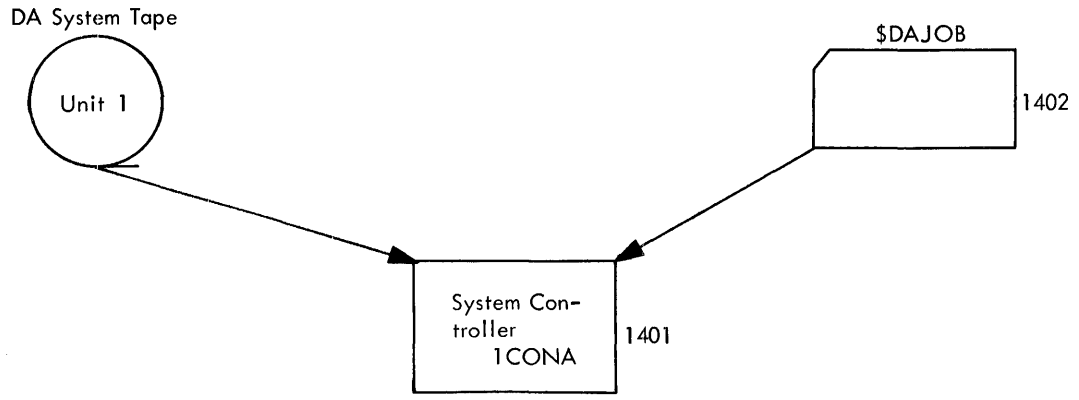
When all operands have been extracted, the switches are checked for blanks and for permissible machine language combinations.

If the $DAJOB card is correct, the next card is read. If that card is a $VERIFY, the first phase of the Verify Program is read in. If it is a $SYSTEM, the System Maintenance is called. Any other card causes the Update Program to be read in.

If the first card of the input deck is not a $DAJOB card, a message is printed and card reading is continued until a $DAJOB is sensed or end of file is detected. At end of file, END OF RUN is printed.

If the $DAJOB card is punched incorrectly, the second card is read. If this card is a $SYSTEM, the System Maintenance Program is called. If it is not a $SYSTEM, an error message is printed and the machine halts.

System Flow



DA SYSTEM MAINTENANCE PROGRAM

Maintenance of the DA System tape is performed by a program contained on the system tape. DA System programs may be added, patched, or deleted from the system tape through the use of the System Maintenance Program (6CONA).

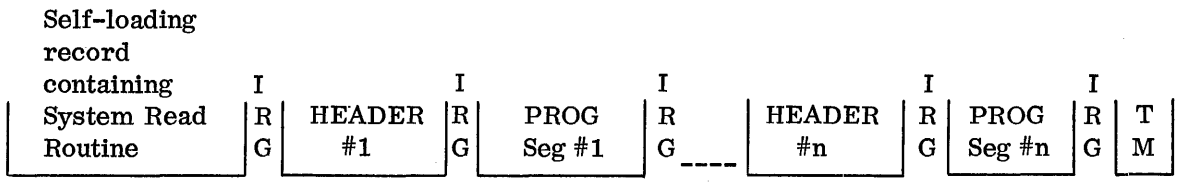
The purpose of the System Maintenance Program is to place DA System records on the system tape in the proper format for the System Read Routine. The System Maintenance Program extracts pertinent information from control cards supplied with the program to be added or patched. The control card information is written on the tape in a header record. The program to be added or patched is then loaded into storage using a modified Autocoder loader and the final coreload record is then written on the tape immediately following its header record.

The System Maintenance Program also has the capability to delete entire programs with their headers, to copy the system tape, and to list the header records.

System Tape Format

The system tape consists of coreload records preceded by their header records. Each program or program segment has a header record and a coreload record. These are placed on the tape by the Systems Maintenance Program.

TAPE FORMAT:



Prog. Seg. Format: Up to 7200 char.

HEADER FORMAT:

Low-Core Address	Starting Address	High Core Address	Program Identification	
3	3	3	5	= 14

where:

low-core address	is the three-character representation of the lowest storage address occupied by the coreload. This value must be greater than 800.
starting address	is the address of the first instruction to be executed.
high-core address	is the highest storage address occupied by the program plus one.
program identification	is the identification of the coreload. All records must have a unique program identification. Program identifications are assigned in ascending collating sequence on the system tape.

System Maintenance Control Cards

The System Maintenance Program is called in the same manner as other DA System programs. A maintenance deck consists of a $DAJOB card, a $SYSTEM card which calls the maintenance program, changes to be made to the system, and finally a $DAEND card.

The format of the $SYSTEM card is:

Cols.	Contents
1 - 7	$SYSTEM

The maintenance process is completed when the $DAEND card is encountered.

The cards between the $SYSTEM and the $DAEND cards are the changes to be made to the system tape.

To prepare a coreload record to be added to the system tape, the user must punch an $ADD control card with the following information and place it in front of the tape Autocoder self-loading object deck.

Cols.	Contents
1 - 4	$ADD
10 - 12	low-core address
13 - 15	starting address
16 - 18	high-core address
76 - 80	program identification

In order to make all programs self-contained, the following instruction sequence should be placed in the front of each source program to be added:

```
DC      @$ADDbbbbb@
DC      +ØRG
DC      +START
DC      +HIGH
```

This sequence will generate the required $ADD card for each record; columns 76 - 80 will pick up the program identification from the job card.

ØRG is the low-core address label

START is the starting address label of the program

HIGH is the high-core address label

HIGH may be defined by including:

```
        LTØRG*+1
HIGH    EQU *+1
        END
```

at the end of the program.

A $ADD card can be used to insert a new record on the system tape or to replace a record with the same program identification.

To delete a program from the tape, the user must punch a $DELETE control card with the following information:

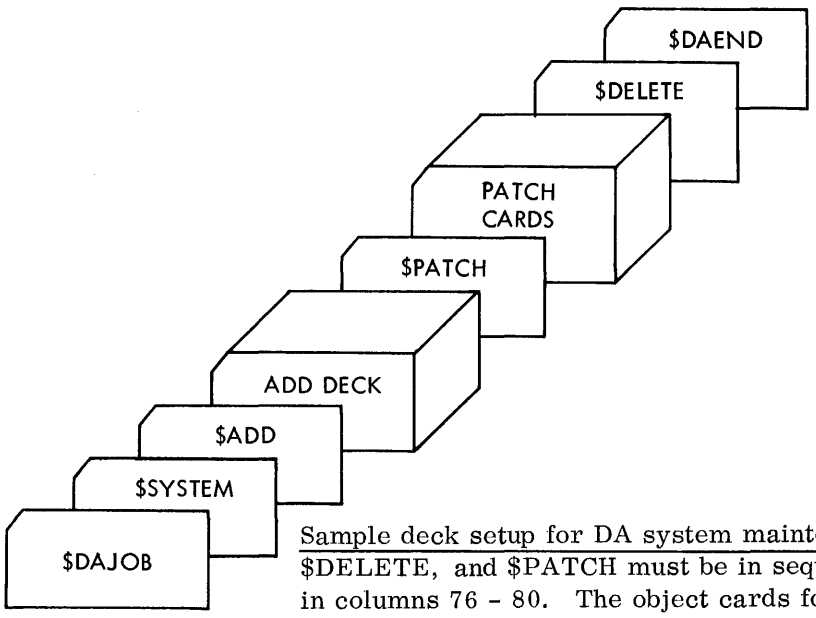
Cols.	Contents
1 - 7	$DELETE
76 - 80	program identification

To patch a program on the tape, the user furnishes the patch cards in tape Autocoder self-loading object deck format preceded by the following control card:

Cols.	Contents
1 - 6	$PATCH
76 - 80	program identification

Addition, deletion, and patching can be performed on the same System Maintenance run if the control cards are in order by program identification. Patching is never performed on a record being added or deleted. A record may be deleted and added on the same run.

To copy the system tape, the user puts no changes between the $SYSTEM and $DAEND cards. Whenever maintenance or copying is performed, the header records are printed.



Sample deck setup for DA system maintenance run. The $ADD, $DELETE, and $PATCH must be in sequence by their identification in columns 76 - 80. The object cards for adding or patching must follow their corresponding $Control card.

Description

The System Maintenance program is executed in two phases. Phase I reads the input file from the card reader, performs diagnostics to check for errors, and prepares the input to the second phase. Phase II reads the tape written during phase I and performs the operations requested in the control cards.

Phase I

A card is read, moved to the print area, and checked for a $ in column 1. If there is no $, an error message is printed and the next card is read. If the card has a $, it is checked to see whether it is a $ADD, $DELETE, $PATCH, or $DAEND. If it is none of these, an error message is printed and the next card is read.

If the card is a $ADD, the addresses in columns 10 to 18 are checked for blanks, zones in the tens position, and to see that they are within the maximum and minimum limits. The ID in columns 76 to 80 is checked for validity and sequence. If any of these errors are detected, the appropriate message is printed; whether or not there are errors, the WRITE3 routine is entered.

In WRITE3, the record is written onto the work tape, and, depending upon the type of control card being written, control is turned back to the control card or load card read routine.

If the control card is a $PATCH, the path followed is the same except for two routines. Since there is no address used in the $PATCH card, there is no address checking. Control information is extracted from the program header on the system tape as the tape is being checked for the presence of the record.

When a $DELETE card is processed, the only checking done is the validity and sequence of the ID and the presence of the program on the tape.

The $DAEND card requires no checking, so it enters a routine that writes the ending record on the work tape and initializes for Phase II.

After a $PATCH or $ADD card is processed, the routine to read and check load card, READLD, is entered. After each card is read, it is checked to see whether it is a system control card, which would terminate loading, a clear storage or bootstrap card, whose presence is indicated on the printer, or an end or execute card, which would also terminate loading. When loading is terminated, control is returned to the control card read routine.

If the card is none of these, it is checked for groupmark/wordmarks, word separators, load address outside the range of the program, and erroneous load instructions. Any of these

errors are noted on the printer. If the load card passes all the checks, the loader instructions are modified to adapt them to tape, and the image is written onto the work tape. Control is returned to the load card read routine until a card is read that causes termination.

$DAEND card causes control to be given to the REWIND routine, which tapemarks and rewinds the work tape, checks to see whether any diagnostic errors were detected, and loads Phase II from its hold area to low core, where it is executed.

If any errors were detected, the machine halts after printing a restart message.

Phase II

The work tape is read in and the mode switch set from the control record. An A indicates add mode, P indicates patch, D indicates delete, and C indicates copy. (The $DAEND causes the mode switch to be set to C, so the remainder of the system is copied without modification. If the first card is the $DAEND, the whole system is copied.)

The old master system tape is composed of header records and master records written alternately. A switch is maintained so that the composition of the next record on the tape is known at any time.

As the first header is read, its ID is compared with the ID in the change file record from the work tape. An equal compare indicates that the program to be modified has been located. If in add mode, the header and program record are skipped on the old master, and the new header and program record are written on the new master. The header information is extracted from the change file record, and the bootstrap routine in the control record is branched to begin the building of the program record in core storage. This bootstrap routine reads in the next record from the change file, which is a load record. This load record operates in the same manner as the normal Autocoder card load record, except that when it has completed loading and setting wordmarks, it reads the next record from the change file. This process continues until an end record is executed. This end record was written on the change file as a result of detecting an end card, execute card, or system control card in phase I. It causes loading to be terminated and the program to be written onto the new master tape.

If a high compare results while reading the old master tape, the process is the same for add mode, except that no skipping is done on the old master.

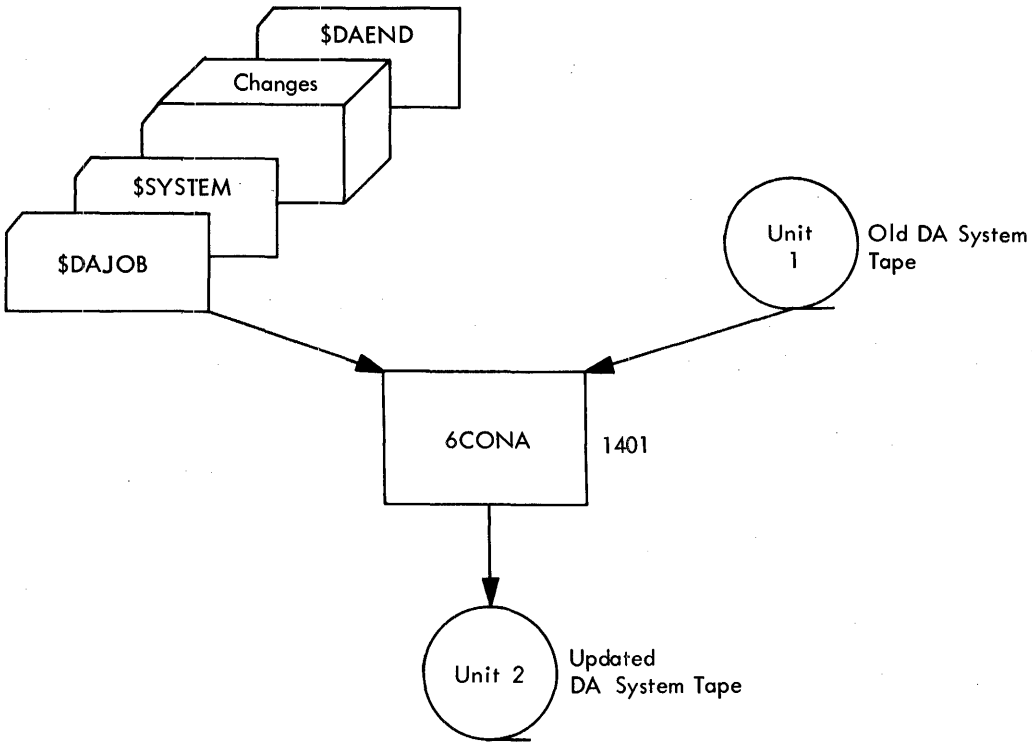
In patch mode, the process is again the same, except that the old program is read into core before loading is begun from the change file.

In copy mode, each header and program record is copied to the new master.

In delete mode, the copy process is duplicated, except that programs to be deleted are not copied.

When end of file is sensed on the old master, the new master is tapemarked, all tapes are rewound, and a list of the headers on the new master is printed.

System Flow



PROGRAM MODIFICATION AIDS

General Modification Aids

The modular system design of the DA System enables the user to readily modify any section. This section contains information to assist the user in making modifications.

Programs under control of the DA System use the following areas of storage, as indicated:

- 01 -- 80 -- are used as a card read-in area
- 81 -- 86

  Position 81 is not used by the DA Controller.

Position 82 contains the following code for the corresponding machine punched in the $DAJOB:

0	System/360
1	1401, 1460
2	1440
3	1410, 7010
4	1620
5	705
6	7080
7	7070, 7072, 7074
8	7040, 7044
9	7090, 7094

Position 83 contains the following code for the corresponding language punched in the $DAJOB card:

0	BAL
1	SPS
2	AUTO
3	BASIC
4	MAP
5	FAP
6	SFL
7	FAL

Position 84 is used as a PAGE OVERFLOW SWITCH. It contains a wordmark whenever the standard heading is printed.

Position 85 contains a wordmark whenever an $UPDATE card is encountered. The wordmark signifies that tape unit 3 should be saved. Position 85 contains the tape unit used for Update output, either 2 or 3.

Position 86 of the communications area is not used by the Controller.

- 87 -- 99 (Index registers) can be used, but must be initialized by a housekeeping routine and not at load time. They will not be disturbed between programs.

- 100 (Read/Punch check position) cannot be used.

- 101 -- 180 (punch area) is used to store the page heading data. The first character is blank (101); positions 2 through 9 of the $Control card are moved into 102 to 110 to indicate the program name in the standard heading; 111 - 117 is blank. The next 54 characters (118 - 171) are the program identification taken from the $DAJOB card. The last nine characters (172 - 180) are bbbbPAGEb.

  If any program requires this area for punching or other purposes, the contents are stored in a hold area before use and restored after use.

- 181 -- 195 is used by Controller when calling in a new program segment. If a GM is used in 181, it is cleared before returning to the Controller.

- 196 -- 199 contains page number; will be incremented by one on each EJECT.

- 200 -- 332 is the print area.

- 333 -- 799 (System Tape Read Routine area and Generalized I/O Routine area) cannot be used. Lowest core location for ORG statement is, therefore, 800. ORG statement must be first statement in a program or Autocoder automatically begins in 333.

All groupmarks must be loaded without wordmarks, having the housekeeping routine set the wordmarks in order to prevent premature tape I/O termination during the tape-load process. This restriction eliminates usage of the G operand in the DA statement. (The Controller does not allow groupmarks with wordmarks to be loaded.)

Each program must also clear its GMWM's before calling in the next program for the same reason.

Wordmarks are placed in the following locations at the beginning of a run. Each program in the system must restore them if they are disturbed.

001	High order of read area
087	High order of X1
092	High order of X2
097	High order of X3
101	High order of punch area

No other wordmarks may be left in these I/O and index register areas.

Word separators cannot appear in the object deck. If needed, load as a 5-8 punch and add 0 zone in the program.

The Controller is used to call in a program, program segment, or overlay. The user branches to a different entry point (400) and supplies the five-position identification of the segment wanted.

Example:

```
B       SYSLNK
DCW     @5VERA@
```

This would call in the first pass of the VERIFY program.

The equate statement SYSLNK EQU 400 must be present in the program.

When calling in a program, the Controller searches forward only; the segment called, therefore, must not have been read before.

Input/Output Modification Aids

The Controller is used to perform all I/O functions, tape read/write, card reading, card punching or printing. The system entry points are defined in each program with the following statements:

Statement		Usage
SYSIØ	EQU 500	Tape read/write
READ	EQU 781	Read card
PUNCH	EQU 789	Punch card
PRINT	EQU 747	Print line
EJECT	EQU 704	Eject page and print heading line
MESSG	EQU 660	Printer operator message

To modify the system configuration (for example, to replace the card reader with a fifth tape unit), the DA System I/O routine is replaced by a user-written routine in the resident I/O program of the DA System. The I/O functions and usage are described below:

1. Tape Read/Write. The necessary data is supplied as a five-position constant with a wordmark in the high-order position:

   Position 1 (high-order): Drive number (2, 3, or 4)

   Position 2 - 4: Address of high-order of I/O area

   Position 5 (low-order): Read or Write (R or W)

   Example: 3555 R

   This causes a record to be read from tape 3 into the area beginning in 555.

   The constant is in line after the branch to the entry point.

   The following equate statements are included in each program:

```
IOCON2    EQU    200
IOCON3    EQU    300
IOCON4    EQU    400
```

   The complete entry is assembled as:

```
B         SYSIO
DCW       +IOCON3
ORG       *-2
DC        +IOARA
DC        @W@
```

The origin statement (*-2) causes the location counter in Autocoder to be decremented by two so that the address constant of the I/O area overlies the zeros of the address constant of the tape unit number.

Note: Tape instructions, other than read and write (rewind, backspace, etc.), are done in the individual programs - not by the Controller.

All tape input/output is in move mode (without wordmarks).

2. Card Reading. The user branches to the system entry point:

```
B       READ
```

The Controller reads a card and returns to the next instruction. Testing for last card is not necessary, because the end of the card input to a run is indicated by a $DAEND card.

3. Card Punching. The user branches to the system entry point:

```
B       PUNCH
```

The Controller performs the card punch and returns to the next instruction.

4. Print a Line. The user branches to the entry point:

```
B       PRINT
```

a. The wordmark in 84 is cleared.

b. The line count is incremented by one.

c. When line count exceeds 57, the line is printed -- followed by a branch to eject.

d. Clear storage and return.

Double spacing is effected by an immediate branch to print after return, since the print area is cleared.

If required, LINTOT, indicating the number of lines per page at 659, may be modified. An A-B zone must be over the units position of the two-position LINTOT.

5. Eject and Print Heading

```
A       B       EJECT       will:
```

a. Set a wordmark in position 84.

b. Skip to Channel p on the printer.

c. Move the punch area to the print area.

d. Add 1 to the page count and move it to the print area.

e. Reset the line count.

f. Set to double-space after printing.

g. Print heading line.

h. Clear storage and return.

A branch to EJECT is given before printing the control card image and before starting any new report.

If it is required to print an additional heading line, position 84 is tested for a wordmark to see whether an eject has occurred. This switch is turned off by the next print.

6. Print Message to the Operator. Some messages require operator action and these are produced by:

```
B        MESSG
```

The carriage is restored before and after printing the contents of the print band. The print band is cleared and control is returned to the program.

Dictionary Modification Aids

Much of the DA System processing depends on the content and coding of the assembly language dictionaries contained as separate records on the system tape. It is possible to modify the DA System processing and output by changing the contents of the dictionaries which are assembled by 1401 Autocoder and reside on the system tape in the same manner as all other system records. For each of the assembly languages processed, there are four dictionaries on the system tape.

The first of the four sets of dictionaries is used by the Analysis program. Each dictionary contains the BCD mnemonic of all operations in the assembly language and also one additional character, which is a code used to specify the type of operation. The dictionaries are arranged in ascending 1401 collating sequence order.

The second of the sets of dictionaries is used by the first phase of the Flowchart program. Each dictionary contains a six-character code for each operation in the assembly language. The six-character code controls the processing of each instruction during the first phase of the Flowchart program. The dictionary is arranged in the same order as the first set of dictionaries; that is, for each entry in the Analysis dictionary there is a corresponding entry in the Flowchart dictionary.

The third set of dictionaries is used by the first phase of the Flowchart program to generate the comments to be inserted in each flowchart box. Each dictionary contains strings of characters called comment skeletons, which are used in conjunction with the information in each DA record to form the comments.

The fourth set of dictionaries is used by the Verify program and contains the mnemonic operation code and machine language representation for each entry.

System Records

The first set of dictionaries records 3ANLA (the first pass of Analysis) and consists of:

Record	Description
3ANAE	1401/1440/1460 Autocoder--SPS
3ANAF	1410/7010 Autocoder
3ANAG	1620/1710 SPS
3ANAH	705/7080 Autocoder
3ANAI	7070/7072/7074 Autocoder
3ANAJ	7040/7044--7090/7094 MAP-FAP
3ANAK	System/360 Assembly Language

The second set of dictionaries follows the record 4CHRA (the first pass of phase I of Flowchart) and consists of:

4CHRB	1401/1440/1460 Autocoder--SPS
4CHRC	1410/7010 Autocoder
4CHRD	1620/1710 SPS
4CHRE	705/7080 Autocoder
4CHRF	7070/7072/7074 Autocoder
4CHRG	7040/7044--7090/7094 MAP-FAP
4CHRH	System/360 Assembly Language

The third set of dictionaries follows the record 4CHRV (the fifth pass in phase I of Flowchart) and consists of:

4CHR1	1401/1440/1460 Autocoder--SPS
4CHR2	1410/7010 Autocoder
4CHR3	1620/1710 SPS
4CHR4	705/7080 Autocoder
4CHR5	7070/7072/7074 Autocoder
4CHR6	7040/7074--7090/7094 MAP-FAP
4CHR7	System/360 Assembly Language

The Verify dictionaries follow system record 5VERT and consist of:

5VERU	1401/1440/1460 SPS--Autocoder
5VERV	1410/7010 Autocoder
5VERW	705/7080 Autocoder
5VERX	7070/7072/7074 Autocoder

Analysis Dictionary Format

Analysis dictionary entries consist of a mnemonic operation code and an Analysis code. The operation code, left-justified, and the Analysis code, right-justified, appear in the operand field of the DC statements in 1401 collating sequence.

Sample Source Statement:

```
DC  @ADD__ M@
```

All dictionary entries are six characters long except 3ANAJ and 3ANAK, which contain seven characters.

Sample 3ANAJ Statement:

```
DC  @ ADD __ M@
```

Sample 3ANAK Statement:

```
DC  @ ADD__M_ @
```

A "y" in the last position of the 3ANAK DC statement indicates that indexing is permitted with this entry.

Additions or modifications can be made to this source deck by inserting the correction in the correct collating sequence and reassembling. If the operand field is to be scanned, care must be taken that the format is compatible with the operand scan rules, which are described in the Analysis Program description.

Any operation code not found in the dictionary is assumed to be a user macro and is processed as such.

If the Analysis dictionary is changed, a corresponding change must be made to the Flowchart Pass 1 Dictionary.

Analysis Codes

The Analysis code may be alphabetic, numeric, or special characters. The Analysis code T indicates the statement is to be considered transparent -- that is, that neither the label nor the operands are to be scanned.

Special characters, except for blank, always mean that some exceptional operation is to be performed. They are as follows:

Character	Definition
$	Indicates the beginning of Qualification Mode (for example, SFX, QUAL)
*	Indicates the termination of Qualification (ENDQ)
%	Indicates the beginning of a macro definition (for example, MACRO in 7090 MAP)
	Indicates the termination of a macro definition (ENDM)

Character	Definition
.	Indicates a change from SPS to Autocoder or Autocoder to SPS in 1401 Autocoder programs (ENT)
,	Flags 1401 SPS machine codes L and M.

For all other codes, the zone portion indicates how to scan the operand, and the numeric portion references the flag to be assigned to the statement on the flagged listing, as follows:

Zone	Instruction
No zone	Do not scan operand.
12	Scan the operand and split off the first field.
11	Scan the operand and do not split off the first field.

Character	Description
1	MACRO
2	Input/output
3	Data defining
4	Branch
5	Halt
6	Assembler control
7	(unassigned)
8	(unassigned)
9	Branch in 7070/7072/7074 Autocoder to scan second operand instead of the first
10	(no flag)
11	(no flag)

Flowchart Pass 1 Dictionary Format

The format of each entry in this set of dictionaries is:

```
DCW @ PLTNNN @
```

where:

P is a pass 1 Flowchart processing code. It is used to process specific instructions (or sets of instructions) during pass 1. The codes used for P are:

Code	Usage
0	No pass 1 processing.
1	If the instruction has exactly one operand, generate a GOTO flowchart operation. This is used for instructions such as the 1401/1410 Autocoder H (halt) instruction, which is either halt or halt and branch.
2	If the instruction has two operands, generate a GOTO flowchart operation from the first operand. This is used for 1401/1410 Autocoder instructions such as CS, which may be clear storage or clear storage and branch.
3	Switch operand 1 and operand 2, unless operand 2 is blank. This is used to regularize certain statements so that the target field of 1401/1410 arithmetic instructions (for example, A, S, etc.) is in operand 1. This code also processes chained 1401/1410 arithmetic statements.
4	Switch operand 1 and operand 2. This is used on 1401/1410 move instructions to place the target field in operand 1. This code also processes chained 1401/1410 move operations.
5	This code processes 1401/1410 SBR instructions to determine whether they represent a subroutine entrance.
6	This code is used to generate a GOTO operation. This is used for instructions which unconditionally generate some type of flowchart box followed by a GOTO -- for example, a 7090 TXI instruction which generates a MODFY box followed by GOTO.
7	This code is used for 7040 and 7090 indexable transfer instructions. If the transfer is indexed, the transfer is changed to an EXIT type.
8	This code is used to expand the three-way 7090/7040 jump instructions (for example, CAS) into two DECID operations.

Code	Usage
9	This code is used to process the 1401/1440/1460 instructions W, P, R.
S	This code is used for the 7080 to switch operand 1 and operand 3.
T	This is used for certain 1401/1410 I/O instructions to distinguish between reading and writing on the basis of the d-modifier.
U	This is used for 1401/1410 conditional branch instructions which may be chained, for example, BCE.

where:

L is a code which is used to calculate the length of the instruction. This code is also used to determine whether the statement is to be deleted; for example, data defining instructions are deleted during the first pass of the Flowchart program.

T is a code which represents the type of SFL operation to be generated by the instruction. The types are:

Code	
0	BLOCK
1	IO
2	MODFY
3	PREDF
4	DECID followed by a YES
5	DECID followed by a NO and EXIT
6	DECID followed by a NO and SUBRT
7	DECID followed by a YES or NO
8	START
9	ENTER
S	WAIT
T	HALT
U	EXIT
V	GOTO

Code	
W	JOB
X	END
Y	SUBRT

NNN is a three-digit comment code number. This number represents the comment to be generated by pass 5 of the Flowchart program.

Flowchart Pass 5 Dictionary Format

The third set of dictionaries is used by pass 5 to generate the variable field portion (cc 21--72) of the SFL card. The comment code which was extracted from the pass 1 dictionaries represents an entry in a pass 5 dictionary. Each entry in a pass 5 dictionary is a three-character address of the beginning location of a comment skeleton. Comment skeletons are composed of two types of information: control characters and comment words. A comment skeleton may begin with either a control character or comment word. The last character of all comment skeletons must be the control character, blank.

Control Characters. Each control character is a single-digit with wordmark. The control codes are:

0	Substitute Assembly Language Sequence Field
1	Substitute Operand Field 1
2	Substitute Operand Field 2
3	Substitute Operand Field 3
4	Substitute BCD Operation Field
5	Substitute Special Field 1
6	Substitute Special Field 2
7	Substitute Special Field 3
8	Substitute 7080 Register number
9	Substitute Special Field 4
,	Insert comma after following word or character
Blank	Terminate skeleton
+	Insert immediately the following character (special characters or numbers)
-	Backspace the variable field pointer

Note: No blanks, as special characters, can be inserted in a comment skeleton. The control character, blank, terminates the skeleton. All numbers 0 - 9 and special characters (collating sequence up to, but not including, A) are reserved for control characters.

Comment Words. Each word of comment begins with a wordmark. A comment word contains no blanks. Blanks are inserted automatically by the comment-processing subroutine.

Special Usage of Comment Code. The three-character comment code for BLOCK generation may not represent a particular comment skeleton, but rather the way in which the comment is to be formed and the processing to be performed in pass 5.

If the first character of the comment code is a zero, the three digits represent a true comment code. If the virst character is not a zero, the digit represents a mode (that is, a verb) to be entered into the BLOCK.

Character	Mode
1	No mode
2	Compute Mode
3	Edit Mode
4	Move-to Mode
5	Set Mode
6	Reset Mode
7	Zero Mode
8	Shift Mode
9	Clear Mode

The second character is used to determine whether a storage location is changed by the instruction.

Character	Mode
0	No storage locations are changed.
1	Operand 1 changed by this instruction.

The third character is used to perform pass 5 processing on certain instructions. For example, when a 1401 compare instruction is encountered in pass 5, the third character of the comment code specifies that the operands of the compare instruction are to be saved in special fields. At the occurrence of the subsequent test and branch instructions (for example, BH), the saved operands may be printed as part of the comment in the decision box.

Verify Dictionary Format

Each dictionary entry is a nine-character literal defined with a DCW statement. The first three characters are used for the machine operation code key and the last six for its associated mnemonic.

In all dictionaries if a mnemonic code cannot be specifically determined by the three-character key alone, a general mnemonic is inserted to give an indication of the type of operation. This general mnemonic is enclosed in asterisks; therefore, it cannot be more than four characters (three for 1410). Example: @LBR*I/O*-@.

Individual format variations are described for each record.

5VERU. The machine operation code key is OND, where O is the machine operation code.

N is blank for all operations except those with a d-modifier and one or two operands, for which it is an A (one) or B (two).

D is the d-modifier if any exists, blank if it does not. Example: @BASBE_ _ _ _@.

General Mnemonic		Key
I/O	Input/output operation	LBR, LBW, MBR, MBW

A few 1401 operations are tested in VERIFY itself and do not appear in the dictionary. These are BBd, WBd, U** and K** (where ** is variable), which give BCE, BBE, I/O, and I/O respectively.

5VERV. The 1410 key has the same format as the 1401. An exception to the standard format is the d-modifier, which is a word separator (0-5-8 punch). Since the DA System does not allow word separations in a table, 5-8 punches should be put as the d-modifier and an asterisk should be placed in the last position of the entry to signal VERIFY that the d-modifier should be a word separator. This means that the mnemonic code can have only five characters. Example: @DB$^{5}_{8}$MRNWR*@.

A few 1410 operations are tested in VERIFY itself and do not appear in the dictionary. These are BBd, WBd, F** and 2** (Where ** is variable), which give BCE, BBE, CC1 and CC2, respectively.

General Mnemonic		Key
I/O	Input/output operations	MBS, MBC, MBR, MBS, MBW, MBX, LBS, LBC, LBR, BLS, LBW, LBX, UBA
L	Lookup operation	TB, TB7

5VERW. The 705/7080 key is OSU, where O is the machine operation code.

SU is the storage unit associated with the particular mnemonic. If no particular SU is associated, 00 is placed in these positions. Example: @,04LSB_ _ _@.

General Mnemonic		Key
TRA	Transfer operation	I00, O00
I/O	Input/output operation	300, 301

5VERX. The 7070 key format is SOP. S is the sign of the operation (+ or -). OP is the machine operation code. Example: @+23ZA2___@.

General Mnemonic		Machine Op Code(s)
B	Branch on *busy	+51
Q	Inquiry control	+54
PC	Priority	+55
ES	Electric switch control	±61, 62, 63
LN	Stacking latch set on	-61
LF	Stacking latch reset off	-62
DC	Data channel control	+93, 94, 96, 97
SC	Sign control	-03
ASS	Additional storage control	+04
FV	Field overflow	+41
S#	Shift control	+50
S	Coupled shift control	-50

DA SYSTEM RECORD IDENTIFICATION AND FUNCTIONS

ROUTINE	RECORD	FUNCTION
Resident I/O	0SYSR	Perform all System I/O
System Controller -- Program Selector	1CONA	Reads and analyzes the $DAJOB card
Update	2UPDA	Update Program
Analysis	3ANAA 3ANAB	The type of source statement is determined, and the DA format is generated
	3ANAE	1401/1440/1460 Operation Code Dictionary
	3ANAF	1410/7010 Operation Code Dictionary
	3ANAG	1620 Operation Code Dictionary
	3ANAH	705/7080 Operation Code Dictionary
	3ANAI	7070/7072/7074 Operation Code Dictionary
	3ANAJ	7040/7044/7090/7094 Operation Code Dictionary
	3ANAK 3ANAQ	System/360 Operation Code Dictionary Produces Operation Code Frequency Report for System/360 source programs

ROUTINE	RECORD	FUNCTION
	3ANAR	For System/360 input, either the Flagged Listing and an input tape to the Chart program are produced, or an input tape with reference cards is produced for 3ANLA
	3ANAU	The Operation Code Frequency Report is produced for languages other than S/360
	3ANAV	Performs same functions as 3ANAR for languages other than S/360
	3ANA9	Phase I cleanup
	3ANLA	Phase I of sort
	3ANLB	Phase II of sort
	3ANLC	The Cross-Reference Report and an input tape for 3ANLD are produced
	3ANLD	Phase I of sort
	3ANLE	Phase II of sort
	3ANLF	An input tape to the Chart Program is produced, in addition to a flagged listing or a flagged listing with operands
	3ANLG	Phase II cleanup
Flowchart	4CHRA	Phase I. Pass 1. Segments source program
	4CHRB	1401, 1440, 1460 Flowchart Operation Dictionary
	4CHRC	1410, 7010 Flowchart Operation Dictionary
	4CHRD	1620 Flowchart Operation Dictionary
	4CHRE	705, 7080 Flowchart Operation Dictionary
	4CHRF	7070, 7072, 7074 Flowchart Operation Dictionary
	4CHRG	7040, 7044, 7090, 7094 Flowchart Operation Dictionary
	4CHRH	Full OS/360 Assembly Language Operation Dictionary
	4CHRS	Pass 2. Builds label table from procedural statements

ROUTINE	RECORD	FUNCTION
	4CHRT	Pass 3. Expands label table with generated labels
	4CHRU	Pass 4. Derelativizes text using generated labels
	4CHRV	Pass 5. Generates SFL program
	4CHR1	1401, 1440, 1460 Flowchart Comment Dictionary
	4CHR2	1410, 7010 Flowchart Comment Dictionary
	4CHR3	1620 Flowchart Comment Dictionary
	4CHR4	705, 7080 Flowchart Comment Dictionary
	4CHR5	7070, 7072, 7074 Flowchart Comment Dictionary
	4CHR6	7040, 7044, 7090, 7094 Flowchart Comment Dictionary
|	4CHR7	Full OS/360 Assembly Language Comment Dictionary
	4CHTA	Phase II. Housekeeping Record
	4CHTB	Phase II. Pass 1. Scans SFL program
	4CHTC	Pass 2, constructs label table
	4CHTD	Pass 3, searches label table for connector operations
	4CHTE	Pass 4, flags off-page box entrances
	4CHTF	Sorts and prints label table
	4CHTG	Pass 5, prints cross-reference list
	4CHTH	Pass 6, expands and rearranges chart records
	4CHTI	Pass 7, constructs line tables
	4CHTJ	Pass 8, generates flowchart

ROUTINE	RECORD	FUNCTION
Verify	5VERA	Pass 1, Object program to tape
	5VERB	SORT3, pass 1
	5VERC	SORT3, pass 2
	5VERT	Pass 3, mnemonic table lookup and generate report
	5VERU	1401, 1440, 1460 Mnemonic Operation Code Dictionary
	5VERV	1410 Mnemonic Operation Code Dictionary
	5VERW	705/7080 Mnemonic Operation Code Dictionary
	5VERX	7070, 7072, 7074 Mnemonic Operation Code Dictionary
System Controller	6CONA	System maintenance routine

APPENDIX TO PROGRAMMER'S INFORMATION

Sort Program

The DA Sort Program utilizes three tape units and is based on the Fibonacci number series principle. In the Fibonacci series, each entry is equal to the sum of the previous two entries (0, 1, 1, 2, 3, 5, 8, 13, 21, etc.). By placing strings on the input units so that their numbers are adjacent entries in this series, the number of times a record must be passed through during merging is held to a minimum. This is superior to other systems particularly when there are fewer than four I/O units available during merging. The DA Sort Program is used twice in the Analysis Program and once in the Verify Program; it is executed in two phases.

In phase I, the unblocked input file is read in, internally sorted using the insertion technique, blocked maximally for the storage size of the object machine, and written onto the two available output units so that the numbers of strings on the two units are adjacent entries in the Fibonacci series.

In phase II, the two input units are merged onto the output unit. In each merge pass, the number of strings on the input unit with the fewer number of strings (the secondary input) is merged with the same number from the input unit with the larger number of strings (the primary input). At the end of each pass the function of each unit is changed so that for the next pass the primary becomes the secondary, the secondary becomes the output, and the output becomes the primary.

This process continues until there is only one string on each of the inputs. During the last pass, deblocking is performed so that the final string is unblocked.

Phase I

Phase I is performed in two sections. In the first section, the first input record is read and the record length computed. The capacity of storage, in records, is calculated as are the addresses of the three internal storage blocks or buckets. The first record is moved to the first bucket, and all initialization with the computed addresses is performed. The first section is then cleared from storage and section two is begun.

In section two, index register 2 is used to keep track of the low-order position of the last full bucket. Index register 3 indicates the low order of the control field of that bucket. As each input record is read, its sort key is compared with the key of the record in the last full bucket. If the input record is high, it is put into the bucket after the last full one. This results in a low-to-high array. If the input is low, the index registers are decremented by the record length so that comparing is done against the next to last full bucket. This decrementing and comparing is continued until the input compares high or until the record in the first bucket has been compared. At this time, all records higher than the input are shifted to the right, and the input record is inserted.

When the available storage capacity has been filled, the sequenced records are written onto an output tape in three blocks. Each time a block of records is written, the records checked to see whether the output produces a sequence break. If it does, the number of sequence breaks on that unit is checked against the number required to maintain the Fibonacci number series. If the number required has been written, the output goes onto the other tape unit.

This process of internal sorting, blocking and distributing the strings is continued until the end of file is sensed on the input unit. The number of strings on the output unit being accessed is then checked for number series conformity. If additional strings are needed, they are simulated by writing a record with all 9s in the key and decrementing each successive key until the proper numbers have been written. If all of the three block areas are not full, padding records of all 9s in the key are written in core until the block being padded is full. This block and any that precede it are then written. All tapes are rewound and Phase II is read in.

Phase II

Phase II of the sort is also executed in two sections. In the first section, the addresses that vary according to record length are computed and inserted in instructions in the second section. Upon the basis of the number of strings on each tape, the primary input, secondary input, and output units for the first merge pass are determined.

In section two, the first section is cleared from storage, the I/O areas are initialized, and merging begins. This is done by reading one record each from the primary and secondary units. These records are compared and the higher is moved to the output area. The file from which that record was taken is read again until a sequence break on that file occurs. When this happens, the other file is read and put out until a sequence break. The number of breaks that have occurred on each input file is compared with the number of strings on the secondary input. If not equal, the process is repeated. When the number

is equal, a merge pass has been completed. The output tape is tapemarked, the output and secondary input are rewound, the sequence counts are reinitialized so that the old primary becomes the new secondary, etc., and the next merge pass begins.

At the beginning of each merge pass, the sequence counts are compared with 1 and 0. When the primary and secondary are both equal to one, the final merge pass is about to begin. When this condition is recognized, the deblocking and pad record deletion routines are initialized so that the final output will be unblocked.

At the beginning of the final merge pass of an Analysis sort, an overlay is called in which permits report printing simultaneously with deblocking and pad record deletion.

Symbolic Flowchart Program

The flowchart generated by the Symbolic Flowchart Program is a 10 x 3 matrix of boxes printed on two consecutive pages of 1403 standard printer paper arranged in three columns: A, B, and C. Starting at the top, the boxes are sequenced vertically A0 through A9, B0 through B9, and C0 through C9. The Flowcharter generates connecting lines between boxes and on-page and off-page connectors.

Before the output matrix is printed, the Flowcharter prints diagnostics, a label table, and a cross-reference label table. In addition, the user may, at his option, request the Symbolic Flowchart Language program to be printed.

Modes of Operation

There are two distinct processing modes of the Symbolic Flowchart Program: chart mode and commentary mode. The chart mode processes all statements which produce flowchart boxes or connectors. The commentary mode processes narrative statements. The Symbolic Flowchart Program determines the mode in which it is operating by examining the format of each input card image.

Symbolic Flowchart Language Input

The coding form used in writing Symbolic Flowchart Language may be the same form used by the 1400 Series Autocoders. Any similar form may be used -- for example, 7070 or 7080 coding forms.

Chart Mode Card Format

SEQUENCE FIELD (cc 1--5)

This field is used for input sequencing. Any characters may be used which belong to the 1401 character set except a groupmark and a tapemark.

LABEL FIELD (cc 6--15)

Symbolic labels may be from one to ten characters in length. A comma, tapemark, groupmark or embedded blank must not be used within a label. In addition, if the first character is a lozenge, the label is used to produce the desired connection between two blocks; but the lozenge label is not printed on the flowchart as a label of that block. If a

label is prefaced by a blank (indented), the label is printed on the flowchart as a label of that block; but such a label cannot be used as the operand of a logical connector operation.

OPERATION FIELD (cc 16--20)

This field describes the type of flowchart box to be drawn or specifies a logical connection to some other box.

OPERAND FIELD (cc 21--72)

This field contains either comments to be printed inside a flowchart box or, in the case of a logical connector operation, a label. In the latter case, the label must begin in cc 21.

Columns 73--80

These columns are ignored by the Symbolic Flowchart Program.

Commentary Mode Card Format

The sequence field is the smae as in chart mode card format. By placing an asterisk in cc 6, the user indicates to the program that the information in cc 16--72 of this input card is to be printed as commentary information. Information contained in cc 1--15 is ignored. The program, when switching to commentary mode, ejects to a new page before printing.

Control Operations

JOB	The JOB card must be the first card of a Symbolic Flowchart Program. The operand is used as a portion of the page heading.
SKIP	Used only in the chart mode. The operand causes the skipping of a number of sequential chart box locations equivalent to the value of the operand. A SKIP operation occurring following the chart location A6 with an operand of 6 causes the skipping of chart locations A7, A8, A9, B0, B1, and B2. The next flowchart box is placed in B3. The skipped chart locations remain blank.
EJECT	Can be used in either the chart mode or commentary mode. It has the effect of immediately terminating the page in process and skipping to a new page. The

operand, if used, controls page numbering. EJECT has three possible operand configurations:

a. The normal configuration is a blank operand. In this case, the page counter is incremented by one and the next sequential page number is assigned to the next page on which processing commences.

b. If the operand is +nnnn, the numeric value, nnnn, is added to the page counter in place of the normal increment of one, and the new total value is the page number of the next page.

c. If the operand is nnnn, the page counter is reset to this numeric value. It becomes the page number of the next page. Only a numeric value less than or equal to 9999 or a blank is a legal operand for the EJECT operation.

SPACE — Can be used only in the commentary mode. The operand field, if any, is a number specifying the number of lines to be skipped.

END — The END card must be the last card of a Symbolic Flowchart Program.

Chart Mode Operations

BLOCK — Generates a processing box. The operand field is printed as comments in the flowchart box.

MODFY — Generates a program modification box. The operand field is printed as comments inside the box.

PREDF — Generates a predefined process box. The operand field is printed as comments inside the box.

DECID — Generates a decision box. The operand field is printed as comments inside the box.

START ENTER BEGIN WAIT HALT STOP EXIT	Generate a terminal box. The operation code is printed inside the box, along with the operand field, if any. The HALT, STOP, and EXIT operations cause a break in logic.
SUBRT	Generates a striped processing box. The label of the referenced subroutine is the first part of the operand field. The label must begin in cc 21 and be followed by a comma. This label and its page and chart location are printed above the horizontal stripe in the flowchart box. The comments after the comma are printed below the horizontal stripe.
IO	Generates an input/output flowchart box. The operand field is printed as comments inside the box.
NOTE	Occupies one chart location. The operand comments are printed without the circumscribed flowchart box lines.

Logical Connector Operations

A vertical line, representing the normal logic flow, connects sequential flowchart boxes. The sequential logic flow may be altered through the use of logical connector operations. The operand field of all connector operations is a label beginning in cc 21. The label indicates a connection is to be made between nonsequential blocks. Wherever possible, the processor generates connector lines between boxes on the same flowchart page. Horizontal line connections may be made from column A to column B, column B to column C, and column A to column C. In the latter case, this can be accomplished if column B is a blank chart location created by a SKIP operation. Connector lines are never generated upward. Rather, an on-page connector symbol is generated to the right of the box. If on-page and off-page entrances are being made to any box, the appropriate on-page or off-page connector symbol is appended to the left of the box.

A decision box causes two logical connector lines to be drawn; one exits to the right, the other exits downward. A GOTO operation always generates a right exit from a flowchart box. If the processor is unable to draw a connector line to the label in the operand field of a YES, NO, or GOTO operation, an on-page or off-page connector symbol is appended to the right of the flowchart box.

GOTO	The GOTO operation generates a connector to the label specified in the operand. The GOTO operation indicates a break in the normal logic flow of a program.

YES	The YES operation generates a right exit from the decision box to the label specified in the operand. The downward exit from the decision box is implied to be the NO exit.
NO	The NO operation generates a right exit from the decision box to the label specified in the operand. The downward exit from the decision box is implied to be the YES exit.

Symbolic Flowchart Language Restrictions

1. Every DECID operation must be followed immediately by a single YES or NO operation; and, conversely, every YES or NO operation must be preceded by a DECID operation. If the second exit from a DECID box causes a break in logic, a GOTO operation should be used.

2. A GOTO operation may not occur after a logic break operation; for example, two consecutive GOTO operations may not be used.

3. The operation code of a terminal flowchart box is printed inside the box. The operand, if any, must be 13 characters or less, including blanks.

4. The operand fields of the BLOCK, PREDF, NOTE, MODFY, IO, and DECID operations must conform to the following format: the maximum length allowable for any single word in the comment operand is 13 consecutive nonblank characters.

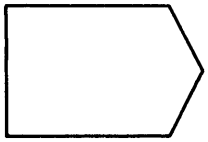
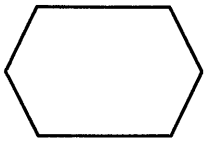
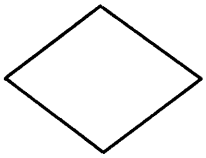
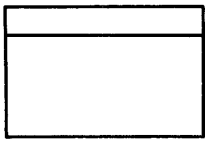
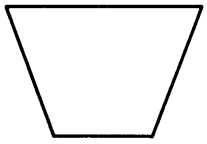
5. The YES, NO, NOTE, SKIP, EJECT, SPACE, JOB, END, and GOTO operations should not contain a label in the label field.

6. Commentary cards may appear only after a break in the program logic flow. An END card or an EJECT card may appear only after a commentary card or a break in the program logic flow. The HALT, STOP, EXIT, and GOTO operations cause a break in the program logic flow.

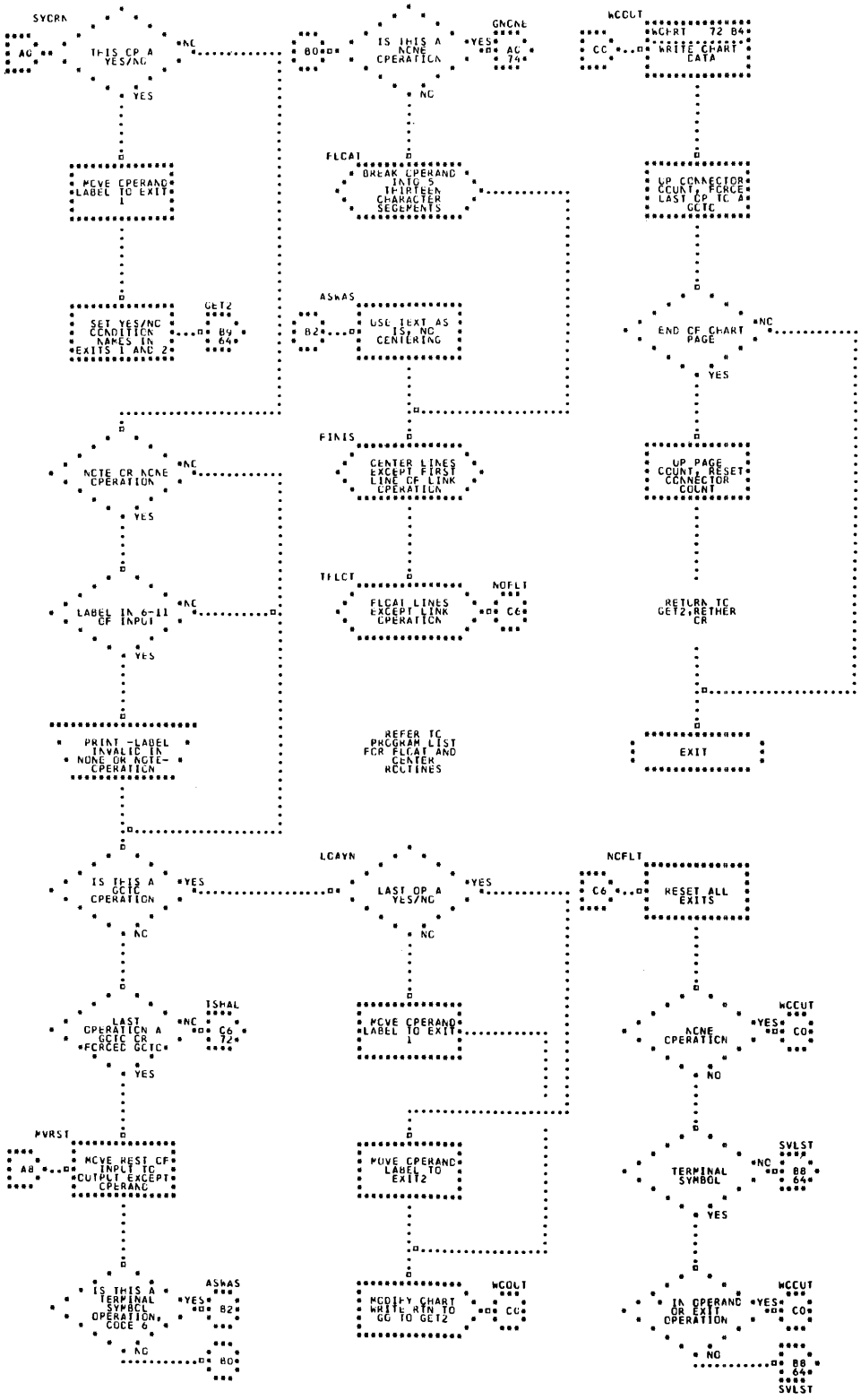
7. The SUBRT operation has a label as its first operand beginning in cc 21 and terminating by a comma. This label is always printed above the horizontal stripe in the flowchart box. The operand field comments following the comma are printed below the horizontal stripe. The maximum length allowable for any single word in the operand comments field is 13 consecutive nonblank characters. The maximum number of allowable comment characters is 24.

8. Skipping is allowed only in the current flowchart column unless the SKIP operation follows a logic break. Skipping is not allowed across a flowchart page.

FLOWCHART BOXES AND CONNECTOR ARROWS GENERATED BY THE SYMBOLIC FLOWCHART PROGRAM

OPERATION	BOX GENERATED	STANDARD BOX NAME
BLOCK		Processing
MODFY		Program Modification
PREDF		Predefined Process
DECID		Decision
START, BEGIN ENTER, WAIT HALT, STOP, EXIT		Terminal
SUBRT		Striped Processing
IO		Input/Output
CONNECTOR		On-Page Connector
CONNECTOR		Off-Page Connector

FLOWCHART OUTPUT MATRIX



OPERATOR'S GUIDE

PROGRAM SETUP

DA System Operation

The following instructions are necessary for operation of the Documentation Aids System:

1. Place the DA System master tape on tape unit 1
2. If tape input (assembly language or SFL), place input tape on tape unit 2
3. Ready tape units 2, 3, and 4
4. Place input card deck on the card reader
5. Turn on I/O CHECK STOP and sense switch A
6. Press CHECK RESET and START RESET
7. Press TAPE LOAD
8. Press START
9. Follow operator instructions on printer
10. A successful run will print END OF RUN and halt with the A and B address registers containing 999.

DA System Maintenance

The following instructions are necessary for the DA System Maintenance run:

1. Place the DA System master tape on tape unit 1
2. Place work tape on tape unit 2 (this will be the new system tape) and tape unit 3
3. Place input card deck in the card reader
4. Turn on I/O CHECK STOP and sense switch A
5. Press CHECK RESET and START RESET
6. Press TAPE LOAD
7. Press START
8. A successful maintenance run will print END OF JOB and halt with the A and B address registers containing 999.
9. File-protect the tape from unit 2 and label it "DA System Tape".

CONSOLE OPERATING INSTRUCTIONS

Each DA System run, whether system maintenance or documentation processing, requires a $DAJOB card as the first card of the card input file, and a $DAEND card as the last.

Several runs may be stacked consecutively in the card reader for continuous batch processing. Each run may require the loading or unloading of tape reels. Instructions to the operator for tape handling will appear on the printer.

HALTS AND MESSAGE LIST

Operator Messages

The following pages indicate all operator messages and instructions. When a halt occurs, the number appears in both the A and B registers.

CONTROLLER OPERATOR MESSAGES

Halt	Message	Record in Which Halt and/or Message Occurs	Explanation
7	None	All records	A system program has issued a call for a program that has been passed or is not on the tape. This is a protected halt. System error.
1	None	All records	I/O routine has attempted to read a tape record ten times. The SELECT light will be lit on the tape unit in which the error occurred. Pressing START will cause the read to be attempted an additional ten times. Replace bad tape and restart run.
999	END OF RUN	1CONA	Completion of DA System run
999	END OF JOB	6CONA	Completion of DA System maintenance run
6	ERROR. TAPE 3 TOO SHORT. REPLACE IT AND RESTART.	6CONA	Reel capacity exceeded

UPDATE OPERATOR MESSAGES

Halt	Message	Record in Which Halt and/or Message Occurs	Explanation
6	ERROR. TAPE 2 TOO SHORT. REPLACE IT AND RESTART.	2UPDA	The computer has sensed an end-or-reel condition on tape 2 during the source card-to-tape operation. Mount a full reel of tape and restart.
6	ERROR. TAPE 3 TOO SHORT. MOUNT NEW TAPE. PRESS START.	2UPDA	While performing the maintenance routine, an end-of-reel condition was encountered.
None	DISMOUNT MASTER TAPE 3. MOUNT SCRATCH.	2UPDA	Self-explanatory
None	DISMOUNT MASTER TAPE 2. MOUNT SCRATCH.	2UPDA	Self-explanatory

ANALYSIS OPERATOR MESSAGES

6	ERROR. TAPE 4 TOO SHORT. MOUNT NEW TAPE. PRESS START.	3ANAA	Reel capacity exceeded
None	DISMOUNT MASTER TAPE 3. MOUNT SCRATCH.	3ANAA	If the input is to be saved, tape unit 3 will unload and this message will be printed.

FLOWCHART OPERATOR MESSAGES

Halt	Message	Record in Which Halt and/or Message Occurs	Explanation
6	ERROR. TAPE "N" TOO SHORT. REPLACE IT AND RESTART.	4CHRA 4CHRU 4CHRV	Reel capacity exceeded. N will be replaced with the corresponding tape unit number.
444	SET UP PRINTER FOR 8 LINES/INCH. HIT START	4CHTA	Use carriage control tape with a punch in channel 1 to allow for 88 lines per page.
None	DISMOUNT MASTER TAPE 3. MOUNT SCRATCH.	4CHTB	Self-explanatory
444	SET UP PRINTER FOR 6 LINES/INCH. HIT START.	4CHTB	Remount normal carriage control tape. May not be required if running stacked CHART jobs.
6	ERROR. TAPE 2 TOO SHORT. REPLACE IT AND RESTART.	5VERA	Reel capacity exceeded

Diagnostic Error Messages

The following pages indicate the DA System diagnostic messages.

CONTROLLER DIAGNOSTICS

Message	Record in Which Message Occurs	Explanation
ERROR. $DAJOB CARD PUNCHED INCORRECTLY. RUN TERMINATED.	1CONA	The operands in the $DAJOB card are incorrect and the next card is not a $SYSTEM. The user must correct the card and rerun.
ERROR. MACHINE AND LANGUAGE COMBO. INVALID. RUN TERMINATED.	1CONA	The assembly language specified in the $DAJOB card may not be used with the machine specified. The user must correct the card and rerun.

Message	Record in Which Message Occurs	Explanation
CORRECT INDICATED ERRORS.	6CONA	During the first phase, a listing of the input deck is printed with any error messages. If any errors do occur, the message will be printed at the end. Error messages which may occur are: 1. ERROR--ADDRESS IN COLS. 10 - 18 TOO LOW. 2. ERROR--I.D. IN COLS. 76 - 80 INCORRECT. 3. ERROR--LOAD INSTRUCTIONS NOT CORRECT. 4. ERROR--ADDRESSES IN COLS. 10 - 18 ILLEGAL. 5. ERROR--HIGH ADDRESS LOWER THAN LOW ADDRESS. 6. ERROR--OUT OF SEQUENCE BY I.D. IN 76 - 80. 7. ERROR--ADDRESS IN COLS. 10 - 18 TOO HIGH. 8. ERROR--NO $ IN COL. 1. 9. ERROR--PROGRAM NOT ON SYSTEM TAPE. 10. ERROR--GROUPMARK WORDMARK LOADED IN XXXX. 11. ERROR--WORD SEPARATOR LOADED IN XXXX. 12. ERROR--NOT A RECOGNIZABLE $ CONTROL CARD. 13. ERROR--LOADING ABOVE $ADD HIGH ADDRESS.

Message	Record in Which Message Occurs	Explanation
		14. ERROR--LOADING BELOW $ADD LOW ADDRESS.
	UPDATE DIAGNOSTICS	
ERROR. BAD DA SYSTEM CONTROL CARD OR INVALID CHARACTER IN COL. 1. RUN TERMINATED.	2UPDA	This message may be caused by incorrect spelling, incorrect format, or an invalid character in an input source language statement.
ERROR. OUT-OF-SEQUENCE CONDITION. RUN TERMINATED.	2UPDA	Out-of-sequence conditions are caused by any of the following: 1. The second parameter of the $DELETE card is less than the first parameter. 2. The first parameter of the $DELETE card is not greater than the sequence number of the last change card. 3. The sequence number of the first change card following the $DELETE card is not equal to or greater than the sequence number of the last change card. 4. The first parameter sequence number specified in the delete control card is not found in the source tape input file. 5. The second parameter sequence number specified in the delete control card is not found in the source tape input file.

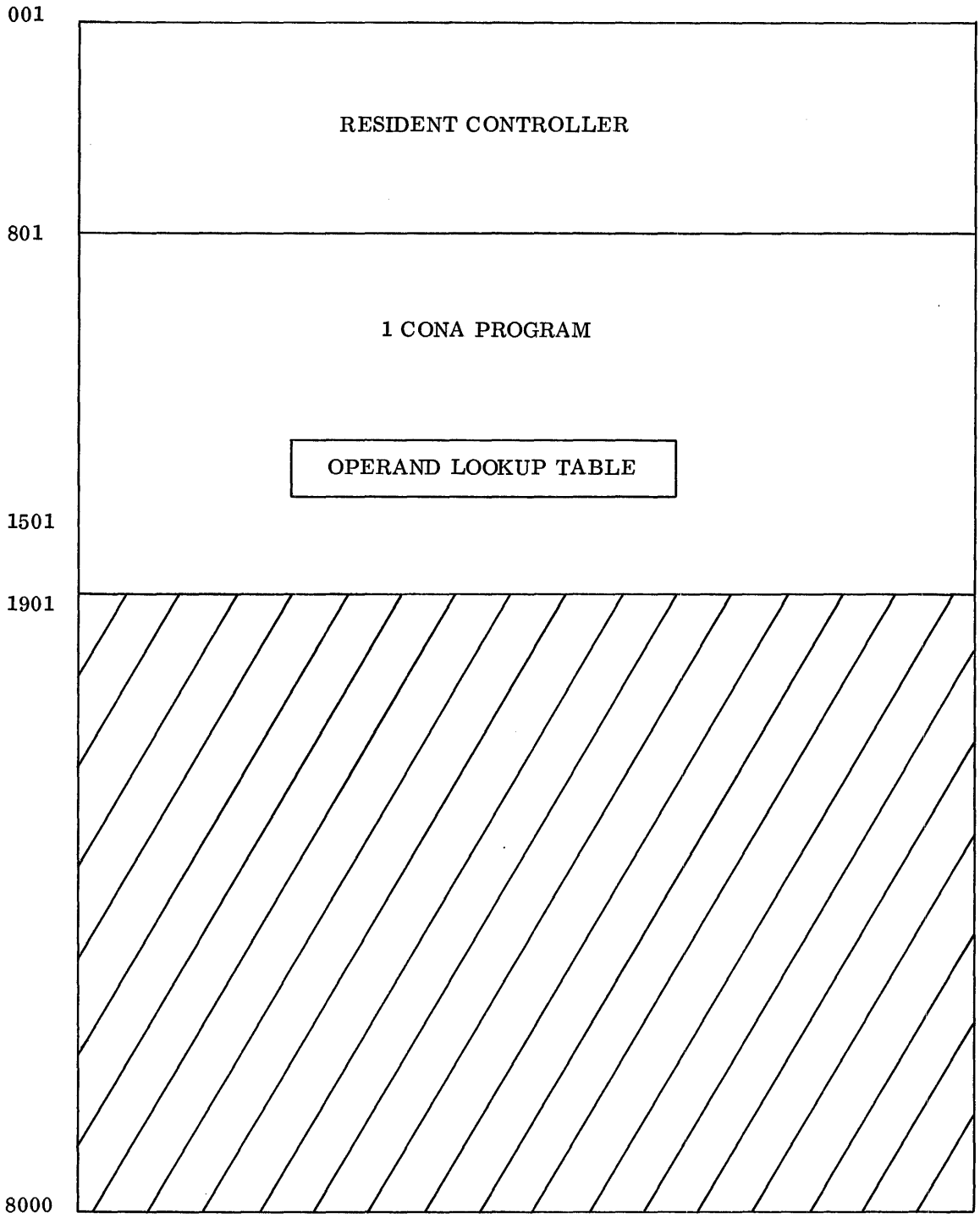
Message	Record in Which Message Occurs	Explanation
		6. The sequence number found in the change input file equals a sequence number in the source input file. In this case, the old sequence number should have been deleted before an addition was attempted.
SEQ ERR	2UPDA	A sequence error has been detected by the program. The out-of-sequence condition is flagged and the run continues.
	ANALYSIS DIAGNOSTICS	
ERROR. ILLEGAL OPTION. RUN TERMINATED.	3ANAA	Optional reports requested on the $ANALYSIS control card cause this halt if options CROSS, OPERAND, and COUNT are punched incorrectly.
ERROR. ILLEGAL CONTROL CARD. RUN TERMINATED.	3ANA9 3ANLG	The $ control card is punched incorrectly.
	FLOWCHART DIAGNOSTICS	
ERROR. CONTROL CARD PARAMETERS UNDERSCORED WITH A 1 ARE IN ERROR.	4CHRA	Self-explanatory
ERROR. END OF FILE ENCOUNTERED WHILE SEARCHING FOR (label).	4CHRA	Segment card label not found
ERROR. SEGMENT (segment limits) CAUSES LABEL TABLE OVERFLOW. PLEASE RESEGMENT.	4CHRS 4CHRT	Table capacity exceeded during Phase I
RUN TERMINATED, INPUT ERRORS.	4CHTB	Violation of Flowchart rules (see "Application Description")

Message	Record in Which Message Occurs	Explanation
INVALID EJECT OPERAND.	4CHTB	Violation of Flowchart rules (see "Application Description")
SPACE OPERATION OVERFLOWS PAGE.	4CHTB	Violation of Flowchart rules (see "Application Description")
NO SPACE OPERAND.	4CHTB	Violation of Flowchart rules (see "Application Description")
LABEL NOT PERMITTED.	4CHTB	Violation of Flowchart rules (see "Application Description")
COMMENTS IN TERMINAL BOX TRUNCATED.	4CHTB	Violation of Flowchart rules (see "Application Description")
INVALID OPERAND	4CHTB	Violation of Flowchart rules (see "Application Description")
INVALID OP	4CHTB	Violation of Flowchart rules (see "Application Description")
BOX COMMENTS TRUNCATED.	4CHTB	Violation of Flowchart rules (see "Application Description")
BOX COMMENTS NOT CENTERED.	4CHTB	Violation of Flowchart rules (see "Application Description")
INVALID PROGRAM LOGIC.	4CHTB	Violation of Flowchart rules (see "Application Description")
INVALID GOTO LOGIC	4CHTB	Violation of Flowchart rules (see "Application Description")
INVALID DECID SEQUENCE.	4CHTB	Violation of Flowchart rules (see "Application Description")
END CARD MISSING, RUN TERMINATED.	4CHTB	END-OF-FILE encountered before END card.
(label) CAUSES LABEL TABLE OVERFLOW.	4CHTC	Table capacity exceeded during Phase II.
(label) IS NOT DEFINED.	4CHTD	The label (in brackets) has not been defined.

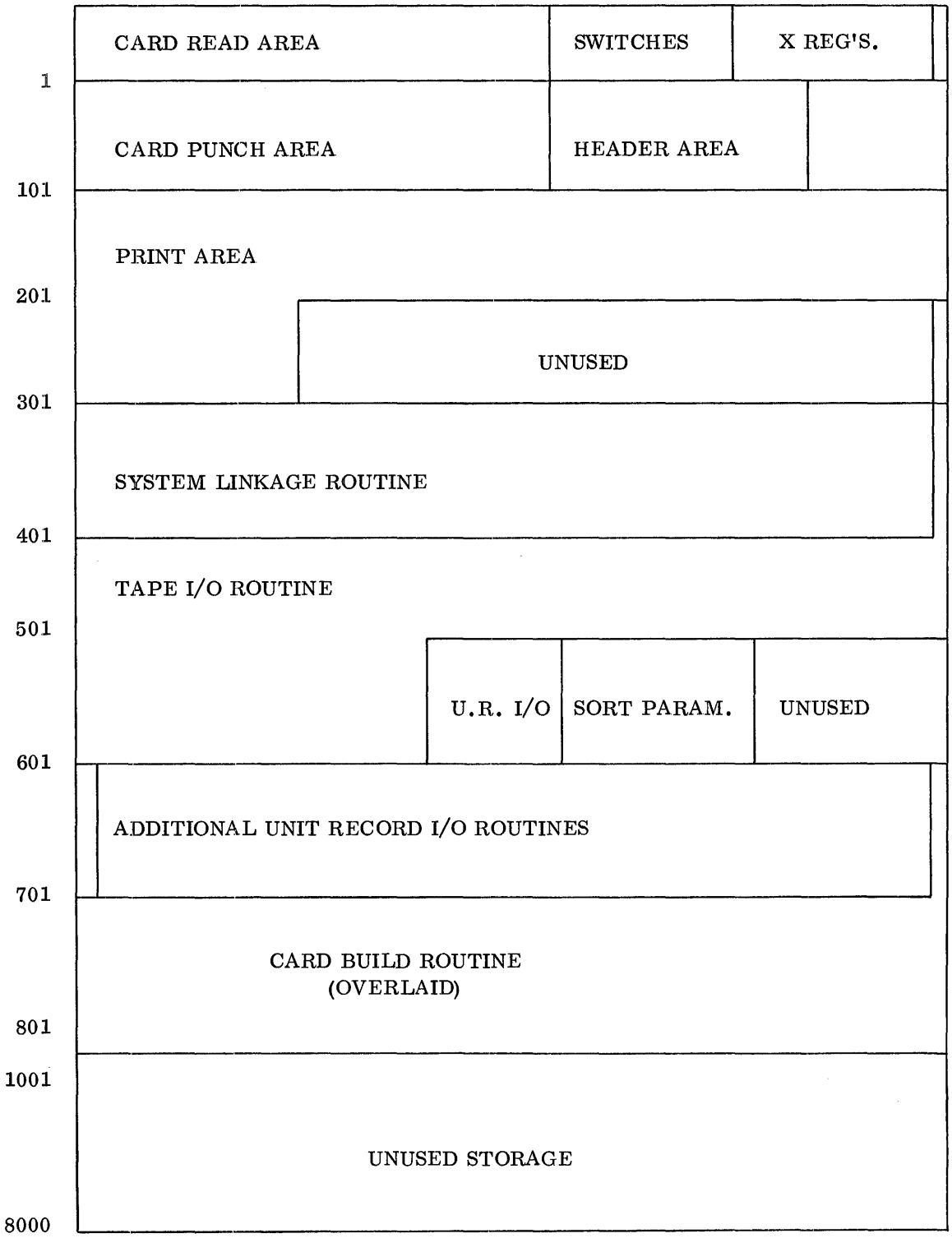
Message	Record in Which Message Occurs	Explanation
(label) IS AN UNREF LABEL.	4CHTE	The label (in brackets) has not been referenced.
MULTIPLY DEFINED.	4CHTF	An identical label has been assigned more than once in the same program.
ERROR. $VERIFY CARD OPTION PUNCHED INCORRECTLY. RUN TERMINATED.	5VERA	Incorrect spelling and invalid language combination are the most frequent errors.
ERROR. MACHINE SPECIFIED ON $DAJOB CARD IS INVALID. RUN TERMINATED.	5VERA	Self-explanatory

STORAGE MAPS

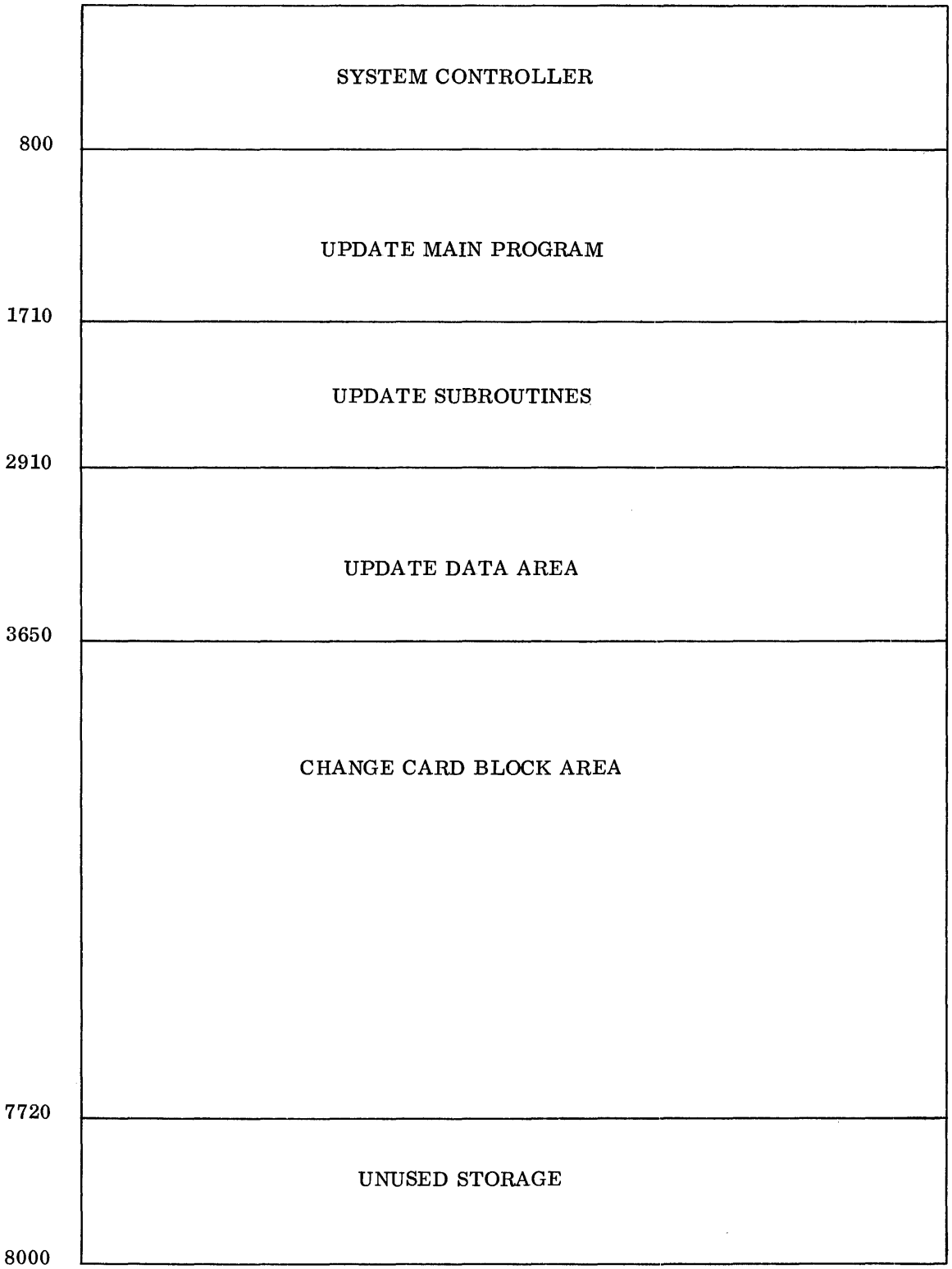
Program Selector



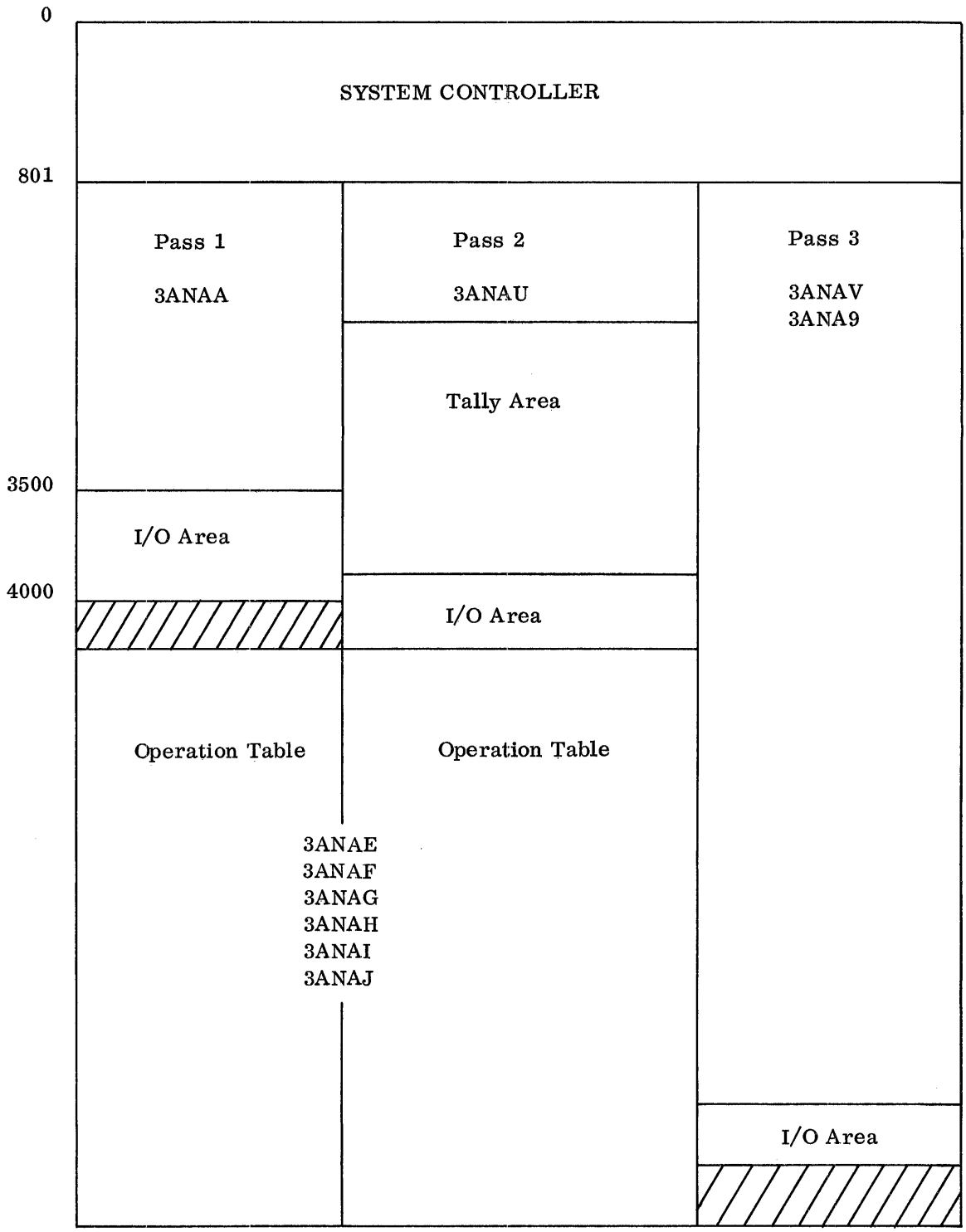
Resident System Controller



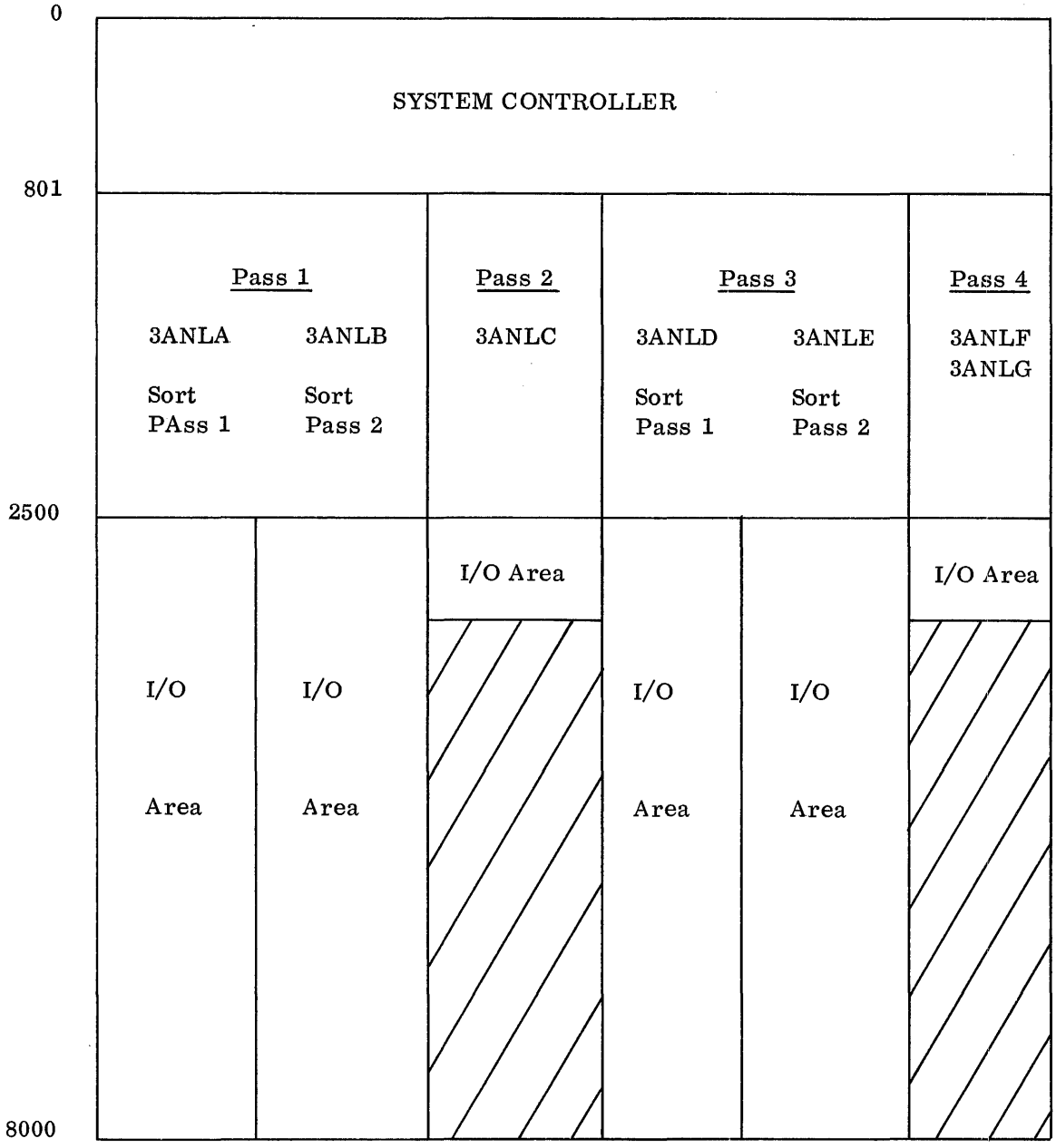
Update 2UPDA



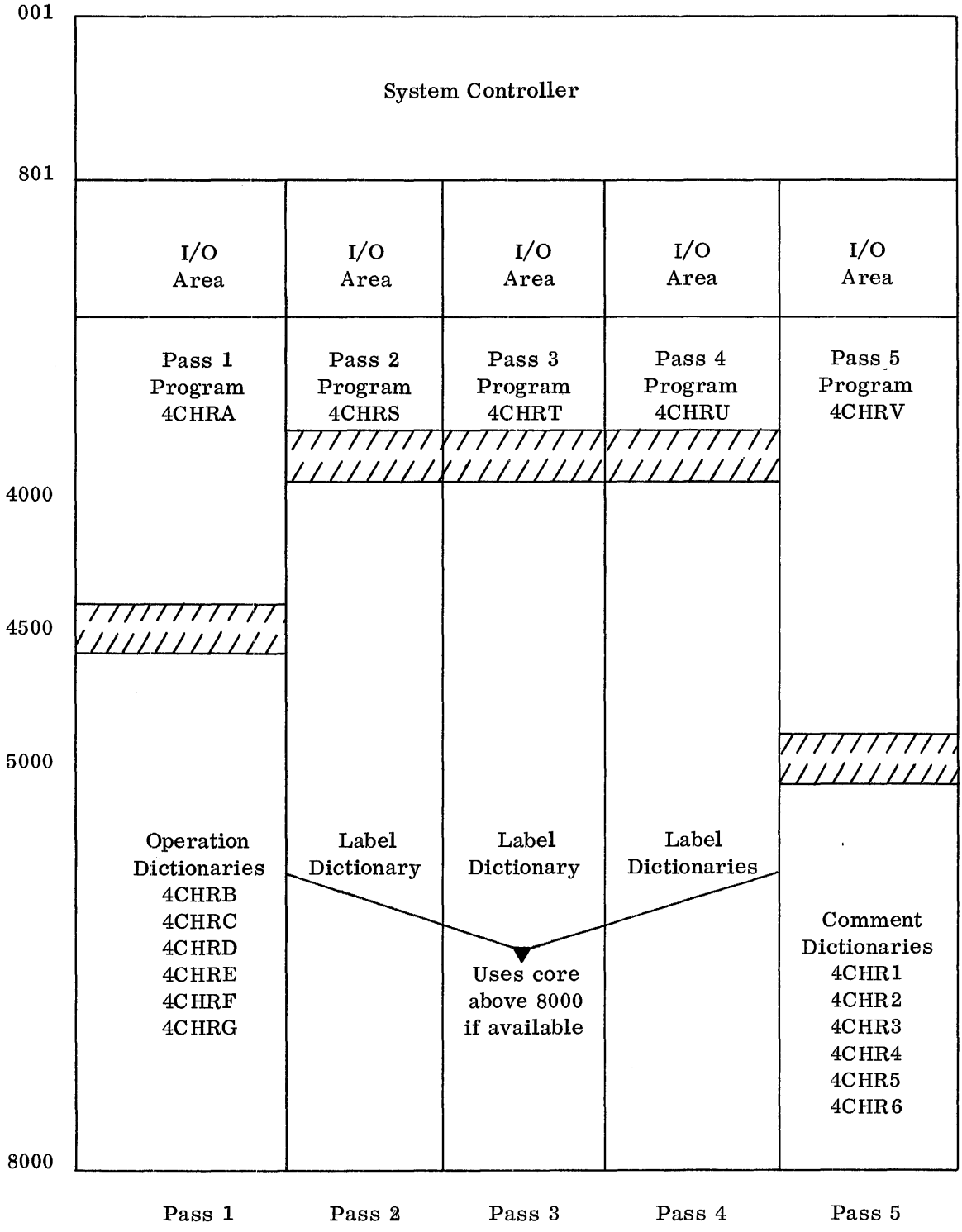
Analysis--Phase I



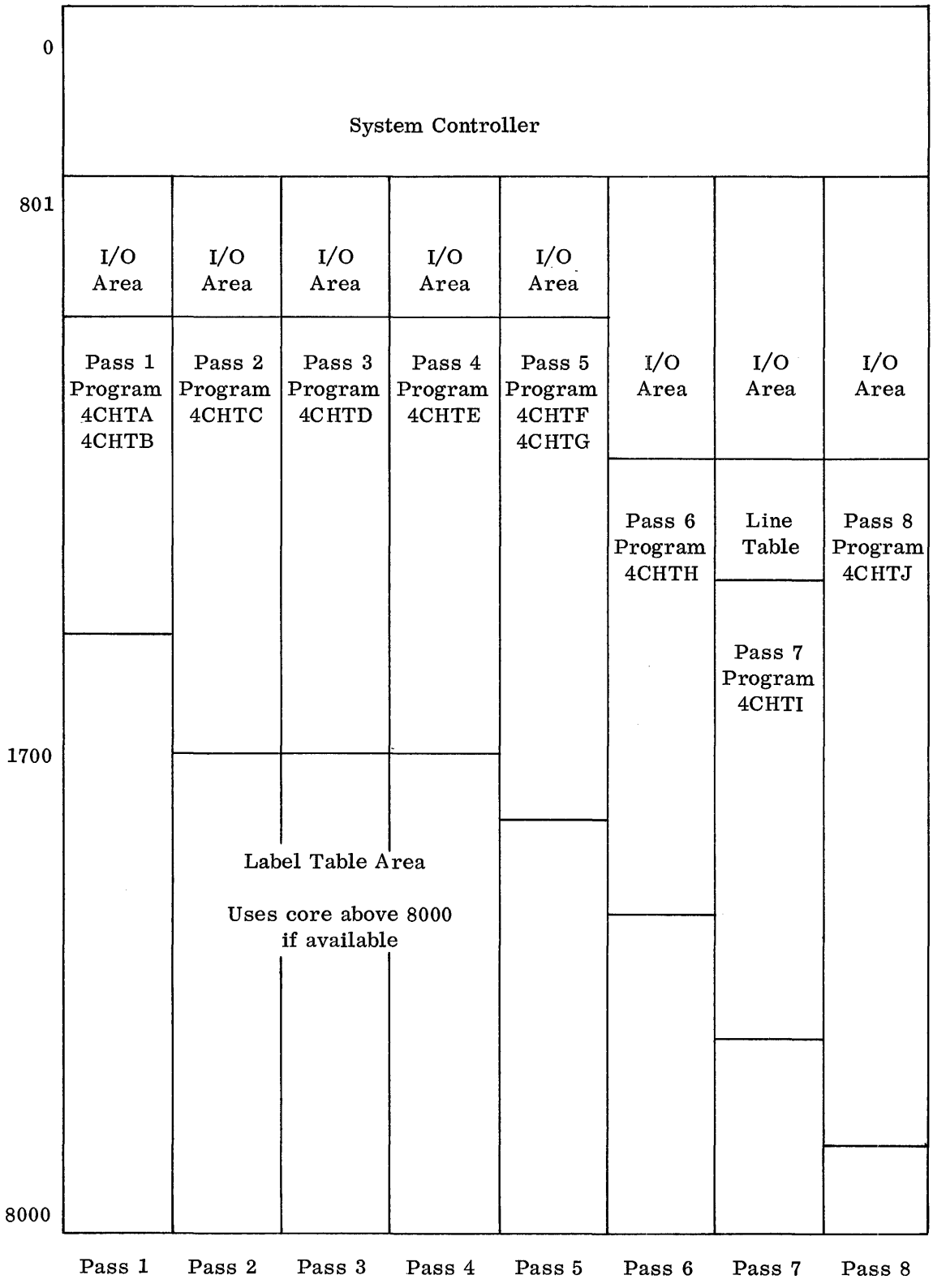
Storage Map of Analysis--Phase II



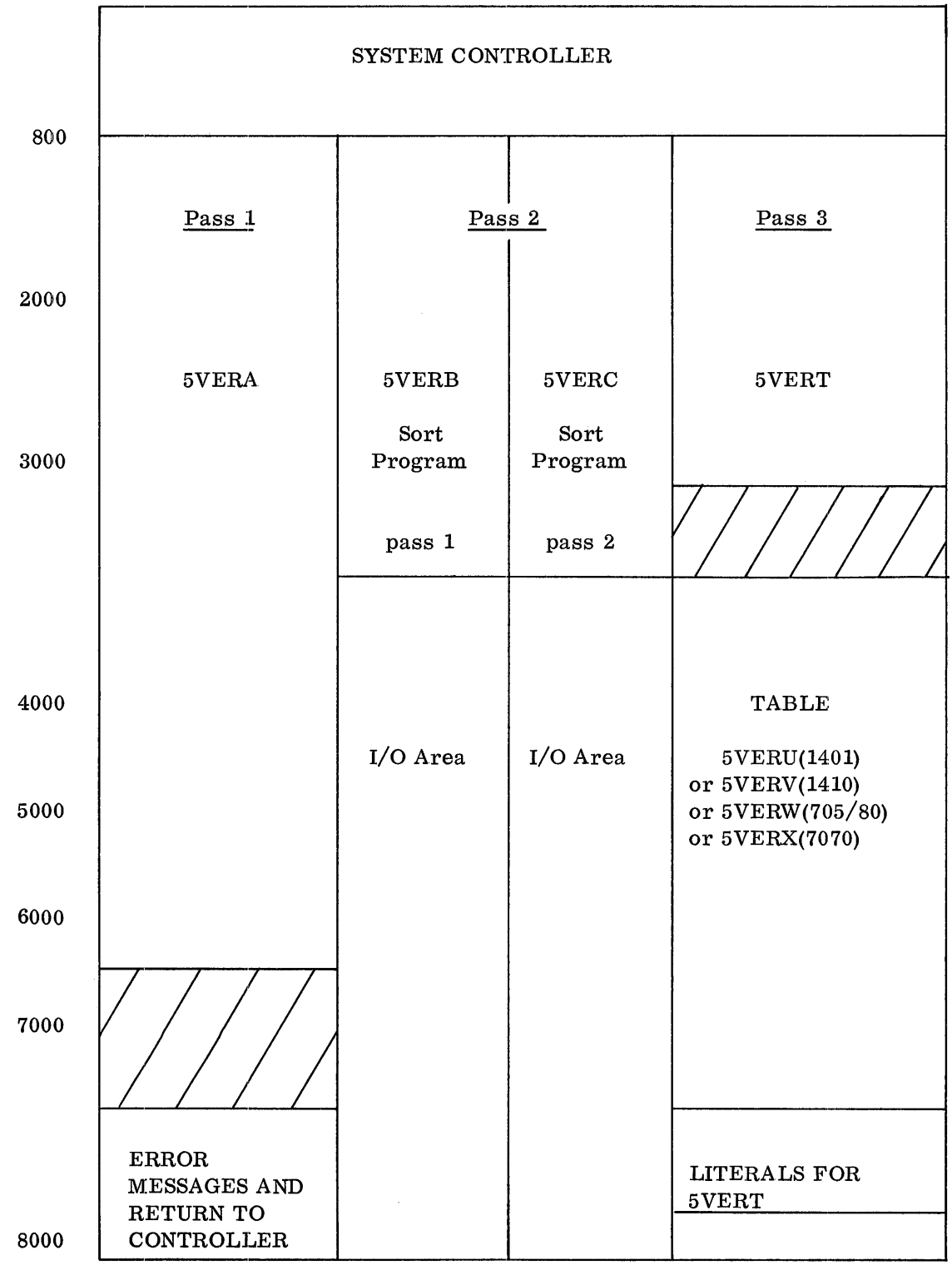
Flowcharter--Phase I

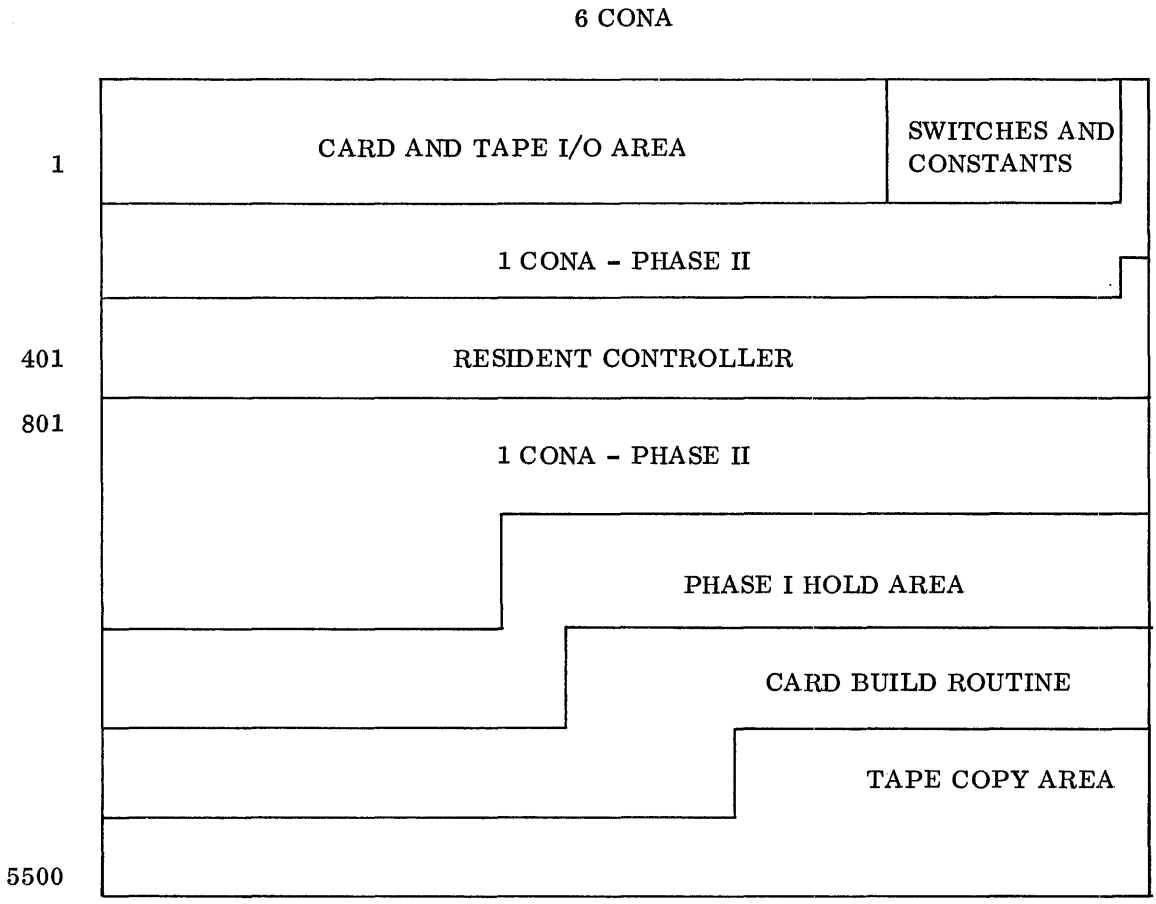


Flowcharter--Phase II



Verify


SYSTEM CONTROLLER
800
Pass 1
Pass 2
Pass 3
2000
5VERA
5VERB
5VERC
5VERT
Sort Program
Sort Program
3000
pass 1
pass 2
4000
TABLE
I/O Area
I/O Area
5VERU(1401)
or 5VERV(1410)
or 5VERW(705/80)
or 5VERX(7070)
5000
6000
7000
ERROR MESSAGES AND RETURN TO CONTROLLER
LITERALS FOR 5VERT
8000



RESTART PROCEDURES

If RESTART is indicated, it must be done from the beginning of the run.

If a significant amount of output has been produced, much of it, if not all, is probably valid. All tapes should be labeled and output returned to the programmer/analyst for review. By deleting and/or changing DA System control cards, rerun time can be held to a minimum.

BIBLIOGRAPHY

Conversion Aids: Documentation Aids (C20-1612), Kingston, New York, 1964.

IBM **Technical Newsletter**

Re: Form No. H20-0177-0

This Newsletter No. N20-0047-0

Date November 15, 1965

Previous Newsletter Nos. None

CHANGES AND ADDITIONS TO PROGRAM REFERENCE MANUAL
FOR DOCUMENTATION AIDS SYSTEM

The attached pages should be inserted into existing copies of H20-0177-0, and the corresponding original pages should be removed and destroyed. Text changes are indicated by a vertical line in the left margin.

Replacement pages are as follows:

Cover
1 - 2
5 - 6
13 - 14
17
17A - 18
31
31A - 32
33 - 50
77 - 78
83 - 84
91 - 94

In addition, the following changes should be made by hand:

p. 4 Add "360" at end of two columns of machine numbers.
p. 10 Change "(FAP/MAP)" to "(FAP/MAP/BAL/FAL)".
p. 25 After "7010" add "or S/360 Model 30"
p. 97 In middle of page, change "smae" to "same"
p. 98 "Chart Mode Operations" should not be underlined.
p. 105 Under "3ANAA" add "3ANAB". (This occurs in two places.)
p. 109 In middle of page, under "3ANAA" add "3ANAB"; under "3ANLG" add "3ANAR".
p. 115 Under "3ANAA" add "or 3ANAB"; after "3ANAU" add "or 3ANAQ"; under "3ANA9" add "or 3ANAR"; under 3ANAJ" add "3ANAK".
p. 117 Under "4CHRG" add "4CHRH"; under "4CHR6" add "4CHR7".

IBM Corporation, Technical Publications Dept., 112 E. Post Road, White Plains, N.Y. 10601

Printed in U.S.A.

Printed in U.S.A. H20-0177-0

IBM®

International Business Machines Corporation
Data Processing Division
112 East Post Road, White Plains, New York 10601