IBM System/3
Disk System Card Utilities
Logic Manual

Program Number:
5702-UT1 Model 10 Disk System

LY21-0523-1
File No. S3-32

Program Product

Preface

This manual is designed to satisfy the documentation requirements of support personnel responsible for maintenance of the IBM System/3 Disk System Card Utility programs. This publication is divided into seven parts – one part for each of the seven programs. Each part contains both general and detailed information. The following sections are included in each part, depending on the size and complexity of the program being described:

1. *Introduction* contains general information about the functions and characteristics of the program.

2. *Method of Operation* describes the data flow and functional flow of the program in general terms, emphasizing the use of data areas.

3. *Program Organization* describes the organization of each routine, using narrative, flowcharts, and diagrams. Flowcharts are designed to provide easy reference to the program listings.

4. *Data Area Formats* describes significant data areas (control blocks, tables, communication area) used by each program.

5. *Object Program* is found in the Sort/Collate section and contains a total description of the Sort/Collate object program, including a sample program dump analysis.

A directory is contained in an Appendix at the back of this publication, giving the entry point and synopsis of each program.

This publication is intended to be a recall mechanism and a debugging tool. In debugging, however, this manual serves best as a guide to the functional sequences of instructions in the program listing.

RELATED PUBLICATIONS

Effective use of this publication requires familiarity with the material in the following publications:

- *IBM System/3 Card and Disk System Components Reference Manual*, A21-9103.

- *IBM System/3 Disk System Operator's Guide*, GC21-7508.

- *IBM System/3 Disk Systems Data Management and Input/Output Supervisor Logic Manual*, SY21-0512.

- *IBM System/3 Disk Systems System Control Program Logic Manual*, SY21-0502.

Second Edition (March 1974)

This is a major revision of LY21-0523-0, and incorporates changes released in Technical Newsletters LN21-7606, LN21-7580, and LN21-7560. A new chapter describing the Gangpunch program has been added. This manual applies to version 10, modification 00 of the IBM System/3 Model 10 Disk System Card Utilities (Program Product 5702-UT1).

Changes are continually made to the specifications herein; any such change will be reflected in subsequent revisions or Technical Newsletters.

Requests for copies of IBM publications should be made to your IBM representative or to the IBM branch office serving your locality.

A form for reader's comments is provided at the back of this publication. If the form has been removed, comments may be addressed to IBM Corporation, Programming Publications, Department 425, Rochester, Minnesota 55901.

FLOWCHARTING TECHNIQUES

Flowcharts in this PLM are identified in the following manner:

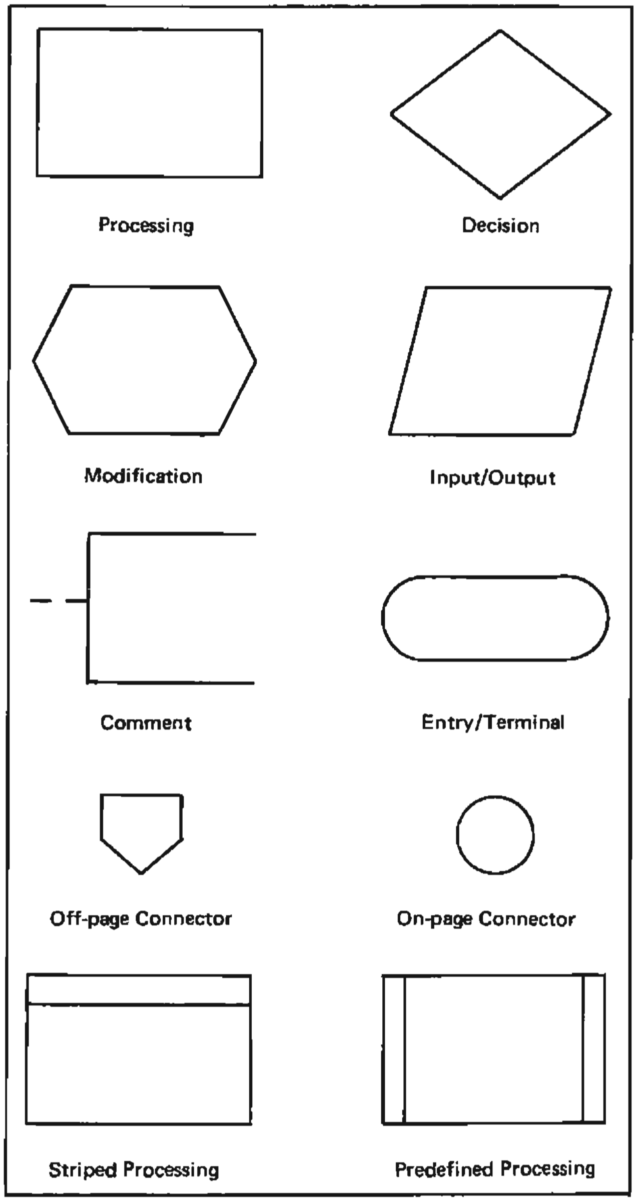
- A flowchart that consists of only one page is identified with a chart ID of: AA

- A flowchart that consists of multiple pages with a chart ID of AA is identified as follows: First page AA-01, second page AA-02 and so on.

- A sequence of flowcharts that are related are identified as follows: First flowchart = AA; second flowchart AB; third flowchart - AC and so on.

- A sequence of flowcharts, each flowchart having multiple pages, are identified as follows: First flowchart with multiple pages AA-01, AA-02, and so on; second chart AB-01, AB-02, and so on; continued through the sequence of flowcharts.

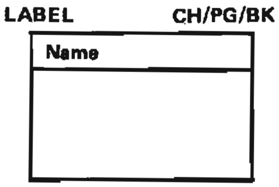
The flowcharting symbols used in this PLM are:



Most of the symbols are self-explanatory but the following two symbols need explanation.

1. The striped processing block indicates the entry of a module or routine that is flowcharted in this PLM.

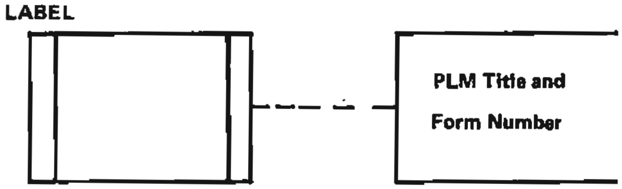
   Example:



CH/PG/BK - Indicates the flowchart, page, and block identification where the module/ routine is flowcharted.

2. Predefined processing indicates a module or routine flowcharted and/or described in another PLM.

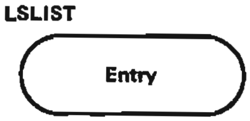
   Example:



Off-page connectors use the CHART/PAGE/BLOCK means of identification. On-page connectors refer to a block on the same page.

The label in the upper lefthand corner, just above the entry symbol, is the entry point in the listing for that part of the program.

Example:



Contents

PART I. 96-LIST 1-1
SECTION 1. INTRODUCTION 1-1
System Requirements 1-1
SECTION 2. PROGRAM ORGANIZATION 1-2
96-List 1-2
SECTION 3. DATA AREA FORMATS 1-4
Read Buffer/Work Area – LSBF1 and LSBF2 1-4
Halt/Syslog Message Table 1-4
Copyright 1-4

PART II. 96-96 REPRODUCE AND INTERPRET 2-1
SECTION 1. INTRODUCTION 2-1
System Requirements 2-1
SECTION 2. METHOD OF OPERATION 2-2
Interpret Only 2-2
Reproduction Functions 2-2
SECTION 3. PROGRAM ORGANIZATION 2-3
96-96 Reproduce and Interpret 2-3
SECTION 4. DATA AREA FORMATS 2-6
Save Area – RWORK 2-6
Reformat Table – RTABL 2-6
Logical Record Area – RBFOUT 2-6
Read I/O Area – RDIO1 2-6
Punch I/O Area – RPCIO1 2-6
Halt/Syslog Message Table – RHLTB 2-6
Buffer Associated IOBs – RPCIB1, RDIB1 2-6
Define the File – DTF 2-6
Print I/O Area – RPRTIO 2-6
Copyright 2-6

PART III. SORT/COLLATE 3-1
SECTION 1. INTRODUCTION 3-1
System Requirements 3-1
Program Structure 3-1
SECTION 2. METHOD OF OPERATION 3-2
Generation Phase ($CSORT) 3-2
Diagnostics Print Phase (CSPRT) 3-3
SECTION 3. PROGRAM ORGANIZATION 3-4
Generation Phase ($CSORT) 3-4
Diagnostics Print Phase ($CSPRT) 3-6
SECTION 4. DATA AREA FORMATS 3-7
Copyright 3-7
Sort/Collate Interphase Area 3-7
SECTION 5. OBJECT PROGRAM 3-10
Job Modules 3-10
Sort Job Module ($CSSRT) 3-10
Merge Job Module ($CSMRG) 3-11
Match Job Module ($CSMCH) 3-11
Select Job Module ($CSSEL) 3-11
Code Segments 3-12
Branch to Job Module 3-13
Record Identification Code Segments 3-13
Control Word Code Segments (Control Field Card) 3-17
Data Areas 3-20
Sort/Collate Interphase Area 3-20
Control Word Address Table 3-20
Sample Dump Analysis 3-21

PART IV. GANGPUNCH 4-1
SECTION 1. INTRODUCTION 4-1
System Requirements 4-1
Program Structure 4-1
SECTION 2. METHOD OF OPERATION 4-2
SECTION 3. PROGRAM ORGANIZATION 4-9
Phase Descriptions 4-9
Diagnostics Phase (GANGP) 4-10
Decimal to Binary Conversion Routine 4-13
Numeric Field Test Routine 4-13
Master or Detail Record Selector Information Identification Check Routine 4-13
Print a Record Routine 4-14
Print Error Messages Routine 4-14
Execution Phase (GPEXC) 4-15
Update the Counter Routine 4-19
Reset the Counter Routine 4-19
Sequence Check the Match Field Routine 4-19
Determine Record Type Routine 4-20
Build the Output Record Routine 4-20
I/O Interface Routine 4-21
SECTION 4. DATA AREA FORMATS 4-22
Common Area 4-22
Define the File – GPDTF1 (diagnostic phase), PRIDTF and SECDTF (execution phase) 4-29
Error Table 1 – ERTAB1 4-30
FDP Table – FDPADS 4-31
Halt/Syslog Message Tables 4-32
Logical Record Buffer – LRBPRT 4-33
Physical Print Buffer – PPNTBF 4-33
Physical Punch Buffer – PPCHBF 4-33
Physical Read Buffer – PREDBF 4-33
Read Buffer – GPRD1 4-33

PART V. DATA RECORDING 5-1
SECTION 1. INTRODUCTION 5-1
System Requirements 5-1
SECTION 2. METHOD OF OPERATION 5-2
General Flow of the Data Recording Program 5-2
SECTION 3. PROGRAM ORGANIZATION 5-4
Initializing Routine 5-4
Display Routine 5-4
Return Routine 5-4
Interrupt Handler Routine 5-4
Interrupt Service Routine 5-6
Data Key/Self-Check Routine 5-6
Invalid Character Routine 5-7
Skip Key Routine 5-7
Dup Key Routine 5-7
Right Adjust Key Routine 5-7
Release Key Routine 5-7
Field Erase Key Routine 5-7
Error Reset Key Routine 5-8
Record Release Switch Routine 5-8
Record Erase Switch Routine 5-8
Read Key Routine 5-8
Program 1 Key Routine 5-8

Program 2 Key Routine 5-8
Program Load Switch Routine 5-9
Adjust Routine 5-9
End or Beginning of Program Defined Field Routine . 5-9
Test for Auto Skip Field Routine 5-9
IOS Interface Routine 5-9
SECTION 4. DATA AREA FORMATS 5-22
Assembly Area – ASSM 5-22
Hold Area – HOLD 5-22
Print Area – PRINT1 5-22
Print Area – PRINT0 5-22
Binary Column Indicator – BCNT 5-22
Decimal Column Indicator – DCNT 5-22
Column Indicator – LITE 5-23
Stick-Light Table – TAB 5-23
Current Control Area – PROG 5-24
Program 1 – PGM1 5-24
Program 2 – PGM2 5-24
Sense Table – STAT/DATA 5-24
Control Code – CCODE 5-24
Read Buffer -- RDBUF 5-26
Copyright 5-26
MFCU IOCS Parameter List 5-26
Buffer-Associated IOB – PUIOB, RDIOB 5-26
Define the File – DTF 5-26
Data Area Activity 5-28

PART VI. DATA VERIFYING 6-1
SECTION 1. INTRODUCTION 6-1
System Requirements 6-1
SECTION 2. METHOD OF OPERATION 6-2
General Flow of the Data Verifying Program . . . 6-2
SECTION 3. PROGRAM ORGANIZATION 6-4
Initializing Routine 6-5
Column Indicator Display Routine 6-5
Return Routine 6-6
Interrupt Handler Routine 6-6
Interrupt Service Routine 6-6
Data Key Routine 6-6
Invalid Character Routine 6-7
Skip Key Routine 6-7
DUP Key Routine 6-7
Error Code Routine 6-7
Right Adjust Key Routine 6-8
Release Key Routine 6-8
Field Erase Key Routine 6-8
Error Reset Key Routine 6-8
Record Release Switch Routine 6-9
Record Erase Switch Routine 6-9
Read Key Routine 6-9
Program 1 Key Routine 6-9
Program 2 Key Routine 6-9
Program Load Switch Routine 6-9
End or Beginning of Program Defined Field Routine . 6-10
Adjust Routine 6-10
Test for Auto Skip Field Routine 6-10
IOS Interface Routine 6-10
SECTION 4. DATA AREA FORMATS 6-26
Assembly Area – ASSM 6-26
Hold Area – HOLD 6-26
Correct Area – CORRT 6-26
Binary Column Indicator – BCNT 6-26
Decimal Column Indicator – DCNT 6-26
Column Indicator – LITE 6-26
Stick-Light Table – TAB 6-26
Current Control Area – PROG 6-28
Program 1 – PGM1 6-28
Program 2 – PGM2 6-28
Sense Table – STAT/DATA 6-28
Control Code – CCODE 6-29
Remember Switch – REM 6-30
Print Area – PRINT1 6-30
Print 2 Area – PRT2 6-30
Copyright 6-30
MFCU IOCS Parameters – IOB 6-30
Buffer-Associated IOB – PUIOB, RDIOB 6-30
Define the File – DTF 6-30
Data Area Activity 6-30

PART VII. 80-90 CONVERSION 7-1
SECTION 1. INTRODUCTION 7-1
System Requirements 7-1
SECTION 2. METHOD OF OPERATION 7-2
SECTION 3. PROGRAM ORGANIZATION 7-3
Building the Conversion Table 7-3
SECTION 4. DATA AREA FORMATS 7-10
Conversion Table 7-10
Character Table 7-10
Copyright 7-10

APPENDIX: DIRECTORY A-1

INDEX X-1

Section 1. Introduction

The 96-List program is a disk resident program which provides the following functions:

- Reads and counts 96-column cards.
  Prints card count only.

- Reads and counts 96-column cards.
  Single spaces listing.
  Prints card count.

- Reads and counts 96-column cards.
  Double spaces listing.
  Prints card count.

- Reads and counts 96-column cards.
  Triple spaces listing.
  Prints card count.

The user sets the rightmost Address/Data switch on the processing unit console to a specific setting to select the desired program option.

End-of-file is indicated by two consecutive end-of-file cards. The first end-of-file card is printed, but does not terminate the job. The format for the end-of-file card is a /* in columns 1 and 2.

System Requirements

The 96-List program requires:

- IBM 5410 Processing Unit.

- IBM 5203 Printer.

- IBM 5424 Multi-Function Card Unit.

- IBM 5444 Disk Storage Drive.

Section 2. Program Organization

Figure 1-1 shows a storage map of the program.

96-List

Entry Point: LSLIST

Chart: CA

Functions:

- Checks for copyright violation.
- Based on the settings of the rightmost Address/Data switch, the following functions are performed:

Setting	*Function*
0	Reads and counts cards. Prints card count only.
1	Reads and counts cards. Single spaces listing. Prints card count.
2	Reads and counts cards. Double spaces listing. Prints card count.
3	Reads and counts cards. Triple spaces listing. Prints card count.
4–F	(Same as setting 2.)

Exits:

- NCENTR
  1. To load Syslist routine.
  2. To Halt/Syslog routine.
  3. To EOJ transient routine.
- LSEND1 – to Syslist routine.
- DMMFFF – to Full Function MFCU IOS routine.

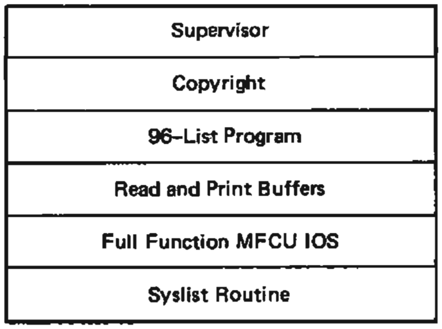
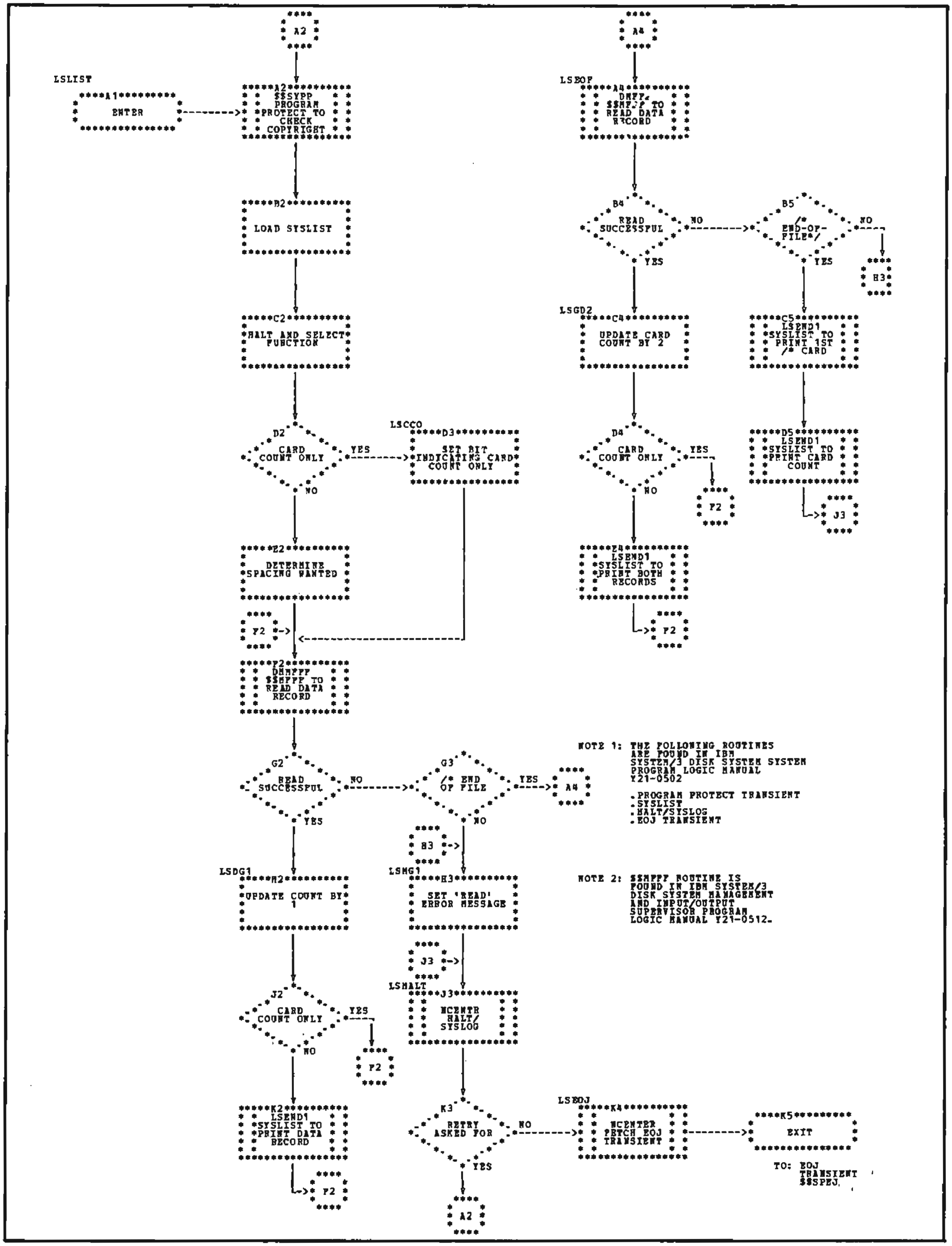


Figure 1-1. Storage Map for the 96-List Program



NOTE 1: THE FOLLOWING ROUTINES ARE FOUND IN IBM SYSTEM/3 DISK SYSTEM SYSTEM PROGRAM LOGIC MANUAL Y21-0502

- .PROGRAM PROTECT TRANSIENT
- .SYSLIST
- .HALT/SYSLOG
- .EOJ TRANSIENT

NOTE 2: $$MFFF ROUTINE IS FOUND IN IBM SYSTEM/3 DISK SYSTEM MANAGEMENT AND INPUT/OUTPUT SUPERVISOR PROGRAM LOGIC MANUAL Y21-0512.

Chart CA. 96-List

Section 3. Data Area Formats

Read Buffer/Work Area -- LSBF1 and LSBF2

These areas are input buffers for reading cards from the MFCU. The address of these areas is passed to the Syslist routine.

Each buffer is 132 bytes long. The last 36 bytes are an extension for larger printers.

Halt/Syslog Message Table

This is a 7-byte area passed to the Halt/Syslog routine. It indicates the type of halt and corresponding action that will be taken.

When the second of two consecutive end-of-file cards is read, a normal halt occurs. When a normal halt occurs, two options exist: re-try and controlled cancel. If controlled cancel is chosen, the EOJ transient routine is called and the job is ended. If the re-try option is chosen, control is returned to the beginning of the program and the program is restarted.

A halt, with immediate cancel as the only option, will occur if there is a printer error or a MFCU read error. An immediate cancel causes the program to be terminated.

Copyright

This 46-byte area contains the program number for this program and copyright information as follows: 5702- UT1ƀCOPYRIGHTƀIBMƀCORPƀ1970. (The ƀ represents a blank.) The remainder of the area is filled with blanks.

Part II. 96-96 Reproduce and Interpret

Section 1. Introduction

The 96–96 Reproduce and Interpret program is a disk resident program which provides the following functions:

- Interprets 96–column cards.
- Reproduces 96–column cards.
- Reproduces and interprets 96–column cards.
- Reproduces and reformats 96–column cards.
- Reproduces, reformats, and interprets 96–column cards.

The user sets the rightmost Address/Data switch on the processing unit console to a specific setting to select the desired program option.

If an option which requires reformatting is chosen, the user must also prepare reformat data cards.

System Requirements

The 96-96 Reproduce and Interpret program requires:

- IBM 5410 Processing Unit
- IBM 5424 Multi-Function Card Unit (MFCU).
- IBM 5444 Disk Storage Drive.

Section 2. Method of Operation

After receiving control, the Reproduce and Interpret program calls a Program Protect Transient routine which checks for copyright violation. Next the program halts displaying 5F to allow the operator to select the desired function. The Address/Data switch setting is sensed and the selected function is performed. Figure 2-1 shows data flow for the Reproduce and Interpret program.

Interpret Only

A data card is read and the data is moved from the read buffer (RDIO1) to the logical output buffer (RBFOUT). The card is then interpreted as the next data card is read. The data cards are read and printed until end of file is reached.

Reproduction Functions

In all reproduction functions, cards are processed in groups of ten. Ten cards are read and moved to a save area and then the ten cards are punched. If reformatting is specified, all cards are reformatted except those with /* in columns 1 and 2; these cards are reproduced in their original format.

If the reformatting option is not specified, a value is used which causes exact reproduction without reformatting. (See *Reformat Table – RTABL* in *Section 4. Data Area Formats*.)

The basic reproducing function is as follows:

1. Read a card into the read buffer.
2. Move the card image to the data save area.
3. Repeat 1 and 2 until ten cards have been read or the second of two consecutive /* cards has been read.
4. Blank out the buffer.
5. Reformat card image into buffer (unless a /* in columns 1 and 2 is read).
6. Punch the card.
7. Repeat steps 4 and 5 until the save area is empty, or the first of two consecutive /* cards has been punched. (The second /* card is not reproduced.)

The added function of interpreting with reproducing is as follows:

1. Interpret as well as punch the card.

The reformat method is as follows:

1. Point to the first entry in the reformat table.
2. Initialize the move instruction using information in the reformat table entry.
3. Move data to buffer.
4. Point to the next entry in the reformat table.
5. Repeat steps 2-4 until the end of the table is reached.

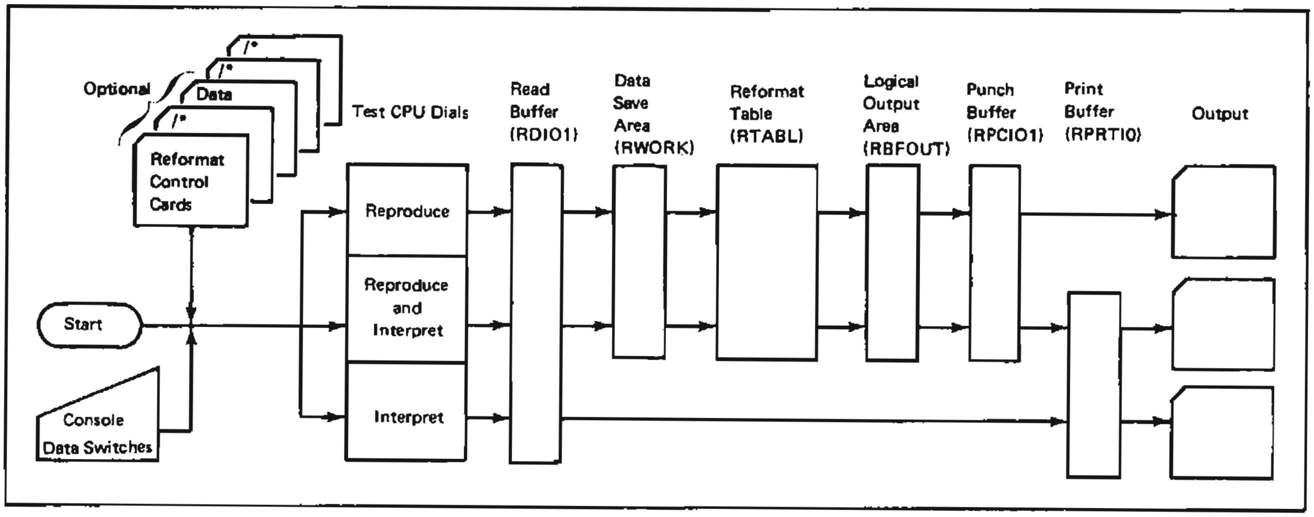


Figure 2-1. Functional Flow of Data for Reproduce and Interpret Program

ART: 55199

Section 3. Program Organization

Figure 2-2 shows the storage map for the program.

96-96 Reproduce and Interpret

Entry Point: REPRO

Chart: DA

Functions:

- Checks for copyright violation.
- According to the setting of the rightmost Address/Data switch, the following functions can be performed:

Setting	*Function*
0	Interpret only.
1	Reproduce only.
2	Reproduce and interpret.
3	Reproduce and reformat.
4	Reformat, reproduce and interpret.
5-F	(Same as setting 2).

Exits:

- DMMFFF – to Full Function MFCU IOS routine ($$MFFF).
- NCENTR
  1. To Halt/Syslog routine.
  2. To EOJ transient routine.

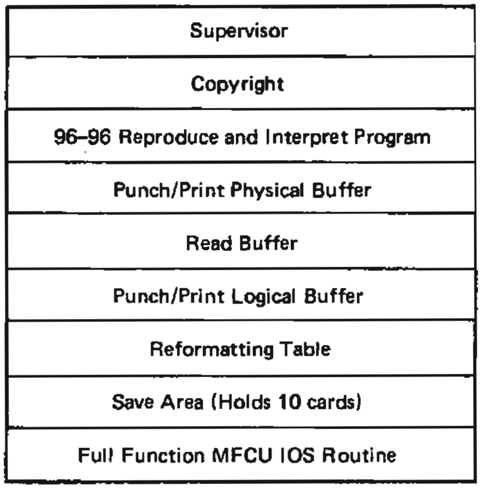

Supervisor
Copyright
96-96 Reproduce and Interpret Program
Punch/Print Physical Buffer
Read Buffer
Punch/Print Logical Buffer
Reformatting Table
Save Area (Holds 10 cards)
Full Function MFCU IOS Routine

Figure 2-2. Storage Map for 96-96 Reproduce and Interpret Program

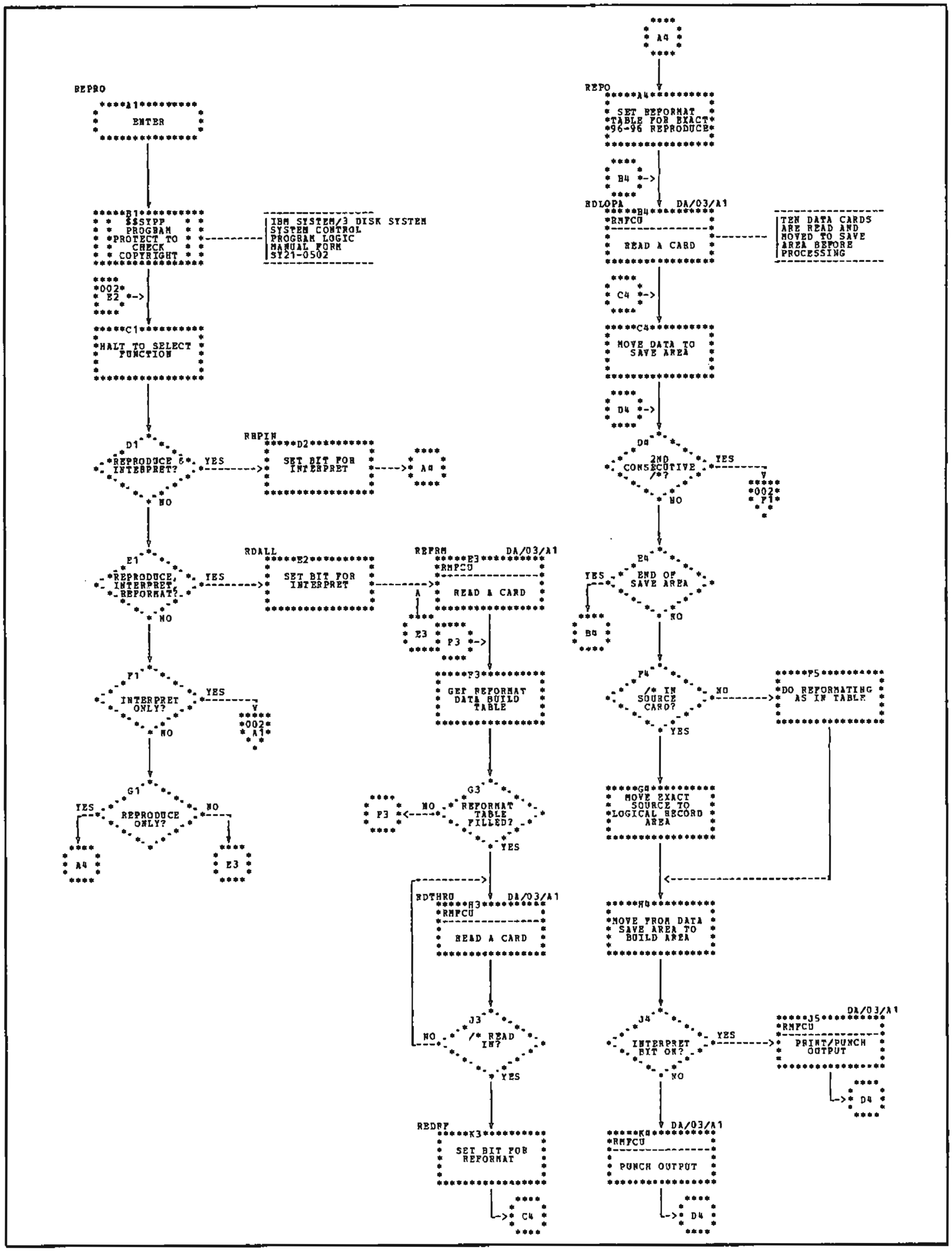


Chart DA. 96-96 Reproduce and Interpret (Part 1 of 3)

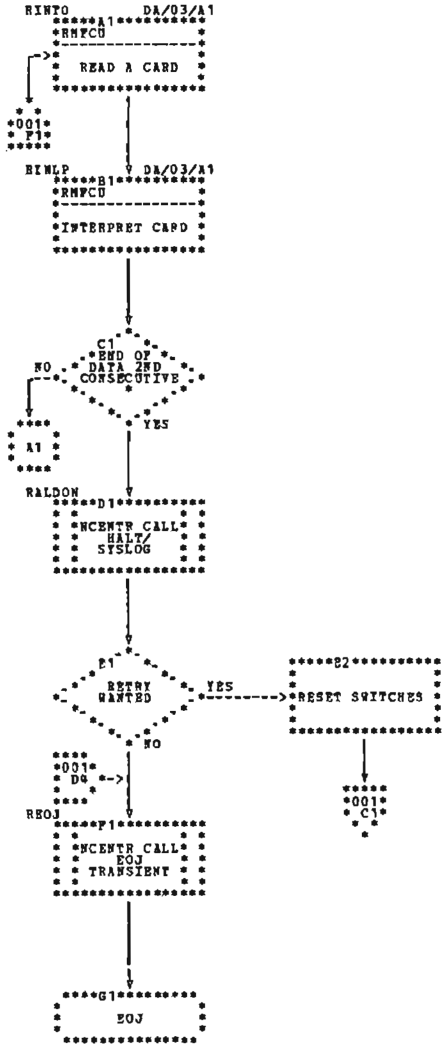


Chart DA. 96-96 Reproduce and Interpret (Part 2 of 3)

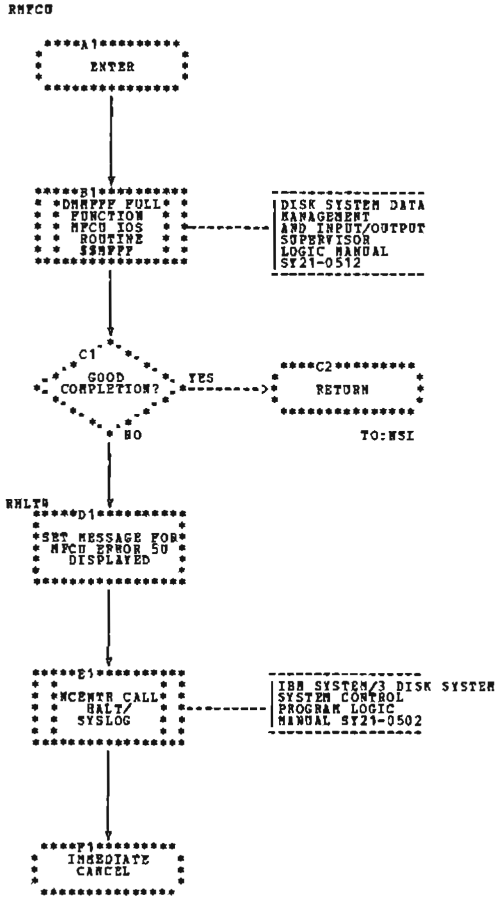

RMFCU
A1
ENTER
B1
DMMFPF FULL FUNCTION MFCU IOS ROUTINE $$MFPF
DISK SYSTEM DATA MANAGEMENT AND INPUT/OUTPUT SUPERVISOR LOGIC MANUAL SY21-0512
C1
GOOD COMPLETION?
YES
NO
C2
RETURN
TO:NSI
RHLT4
D1
SET MESSAGE FOR MFCU ERROR 50 DISPLAYED
E1
NCENTR CALL HALT/ SYSLOG
IBM SYSTEM/3 DISK SYSTEM SYSTEM CONTROL PROGRAM LOGIC MANUAL SY21-0502
F1
IMMEDIATE CANCEL

Chart DA. 96-96 Reproduce and Interpret (Part 3 of 3)

Section 4. Data Area Formats

The areas discussed in this section are used by more than one routine.

Save Area – RWORK

This is a 960-byte area used to store the data from a maximum of ten 96-column cards.

Reformat Table – RTABL

This area can be from 3 to 300 bytes in length. When reformatting is not specified, RTABL has a 3-byte default value of X'005F5F', indicating that one field (cc 1-96) is to be placed (unchanged) in 1-96. The format of each RTABL entry is as follows:

- First byte contains the card column, minus 1, of the leftmost character of the input field to be relocated.
- Second byte contains the card column, minus 1, of the rightmost character of the input field to be relocated.
- Third byte contains the card column, minus 1, of the rightmost character in the reformatted output field.

RTABL is built using the reformatting control cards.

Logical Record Area – RBFOUT

This is a 96-byte area which is used as the output buffer for the logical records.

Read I/O Area – RDIO1

This is a 256-byte area. The Full Function MFCU IOS routine, $$MFFF, uses it as a read buffer.

Punch I/O Area – RPCIO1

This is a 96-byte area. Records that are to be punched are transferred here from the logical record area by the Full Function MFCU IOS routine, $$MFFF. This routine uses the Punch I/O area as a punch buffer.

Halt/Syslog Message Table – RHLTB

This is a 7-byte area passed to the Halt/Syslog routine used to indicate the type of halt and corresponding action, which will be taken.

Buffer Associated IOBs – RPCIB1, RDIB1

See *Part 4. Data Recording, Section 4. Data Area Formats, Buffer-Associated IOB – PUIOB, RDIOB.*

Define the File – DTF

See *Part 4. Data Recording, Section 4. Data Area Formats, Define the File – DTF.*

Print I/O Area – RPRTIO

This is a 256-byte area. Records to be printed are transferred here from the logical record area by the Full Function MFCU IOS routine, $$MFFF. This routine uses the Print I/O area as a print buffer.

Copyright

This 46-byte area contains the program number for this program and copyright information as follows: 5702-UT1ƀCOPYRIGHTƀIBMƀCORPƀ1970. (The ƀ represents a blank.) The remainder of the area is filled with blanks.

Part III. Sort/Collate

Section 1. Introduction

Sort/Collate is a disk resident program which provides the following functions:

- Sorts cards into a sequenced card file.
- Merges two sequenced card files.
- Matches records from two sequenced card files.
- Selects specific cards from a file.
- Sequence checks the card files.

The user supplies input in the form of specification cards which design the Sort/Collate object program to his particular needs.

System Requirements

The Sort/Collate program requires:

- IBM 5410 Processing Unit.
- IBM 5203 Printer.
- IBM 5424 Multi-Function Card Unit (MFCU).
- IBM 5444 Disk Storage Drive.

Program Structure

The Sort/Collate program is comprised of three phases.

1. The Sort/Collate Generation and Diagnostics phase, hereafter referred to as the Generation phase ($CSORT):
   - Reads the specification cards.
   - Diagnoses the specification cards (except the header card).
   - Prints a source listing.
   - Generates the object code.
2. The Diagnostics Error Message Print phase, hereafter referred to as the Diagnostics Print phase ($CSPRT):
   - Diagnoses the header card.
   - Prints error messages for all errors found on the specification cards.
   - Checks the job type and sets switches for later selection of the proper job module.
3. The last phase, the Execution phase, consists of the generated code and the job module selected by the Diagnostics Print phase. The Execution phase processes the user's data according to the generated code. This phase is an object program and is, therefore, discussed in *Section 5, Object Program,* rather than in *Section 3, Program Organization.*

Section 2. Method of Operation

This section describes the general flow of logic and data in the Generation phase and the Diagnostics Print phase. Diagrams are included to convey this logic and data flow. Supporting text is provided as necessary; for the most part, however, the diagrams are designed to be self-explanatory. See *Section 3. Program Organization* for a more detailed explanation of the phases.

Generation Phase ($CSORT)

Figure 3-1 shows the input and output flow for the Generation phase. The Generation phase builds the Sort/Collate interphase area which is used to store information needed for later reference. (See *Section 4. Data Area Formats* for contents of Sort/Collate interphase area.)

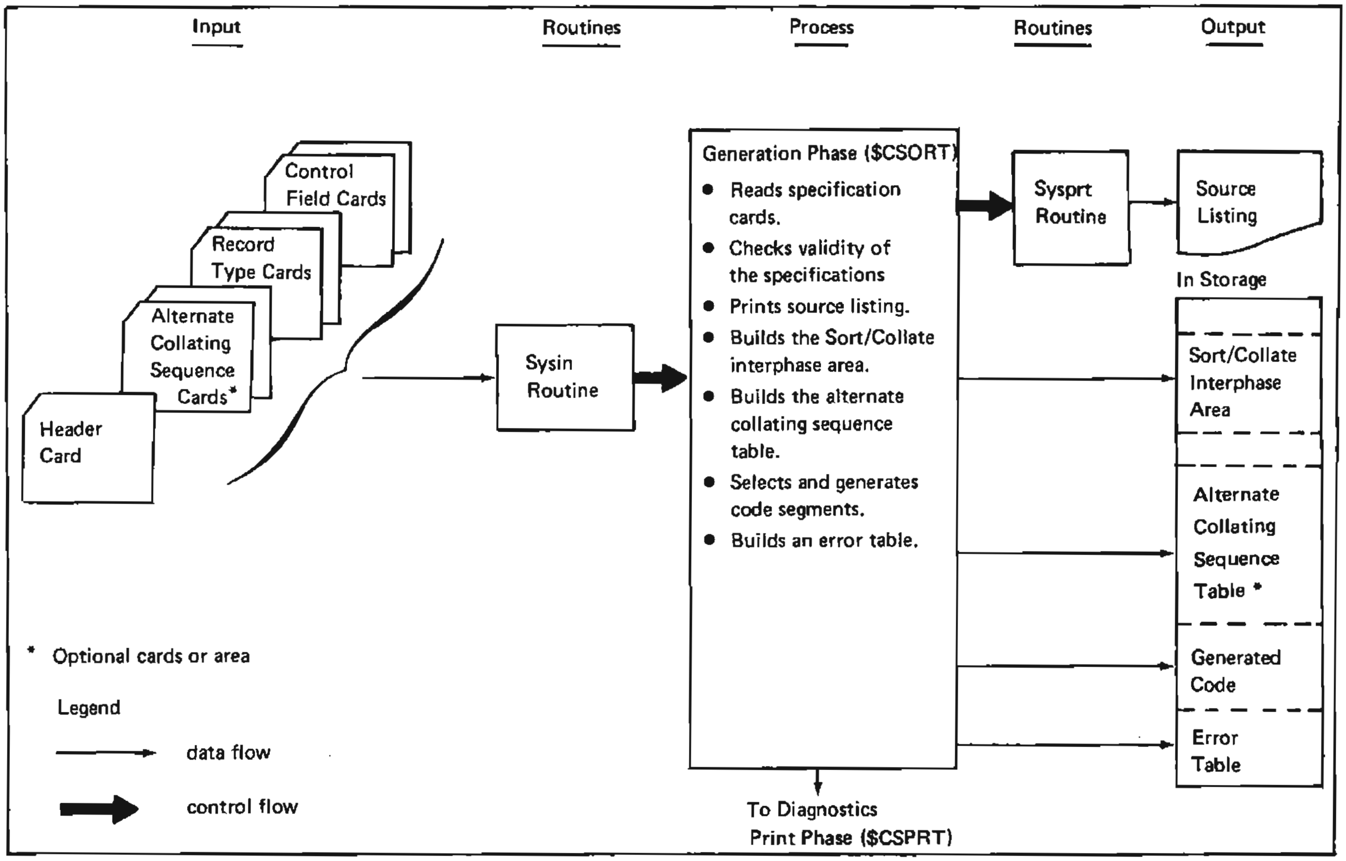

Input
Routines
Process
Routines
Output
Control Field Cards
Record Type Cards
Alternate Collating Sequence Cards*
Header Card
Sysin Routine
Generation Phase ($CSORT)
● Reads specification cards.
● Checks validity of the specifications
● Prints source listing.
● Builds the Sort/Collate interphase area.
● Builds the alternate collating sequence table.
● Selects and generates code segments.
● Builds an error table.
Sysprt Routine
Source Listing
In Storage
Sort/Collate Interphase Area
Alternate Collating Sequence Table *
Generated Code
Error Table
To Diagnostics Print Phase ($CSPRT)
* Optional cards or area
Legend
data flow
control flow

Figure 3-1. Functional Flow of Data and Control for Sort/Collate Generation Phase ($CSORT)

Diagnostics Print Phase ($CSPRT)

Figure 3-2 shows the input and output flow for the Diagnostics Print phase. Terminal errors force an end-of job halt which must be corrected before operation can continue. Warning errors also cause a halt; however, operation can be resumed by pressing the START button (or HALT RESET if you have the Dual Programming Feature).

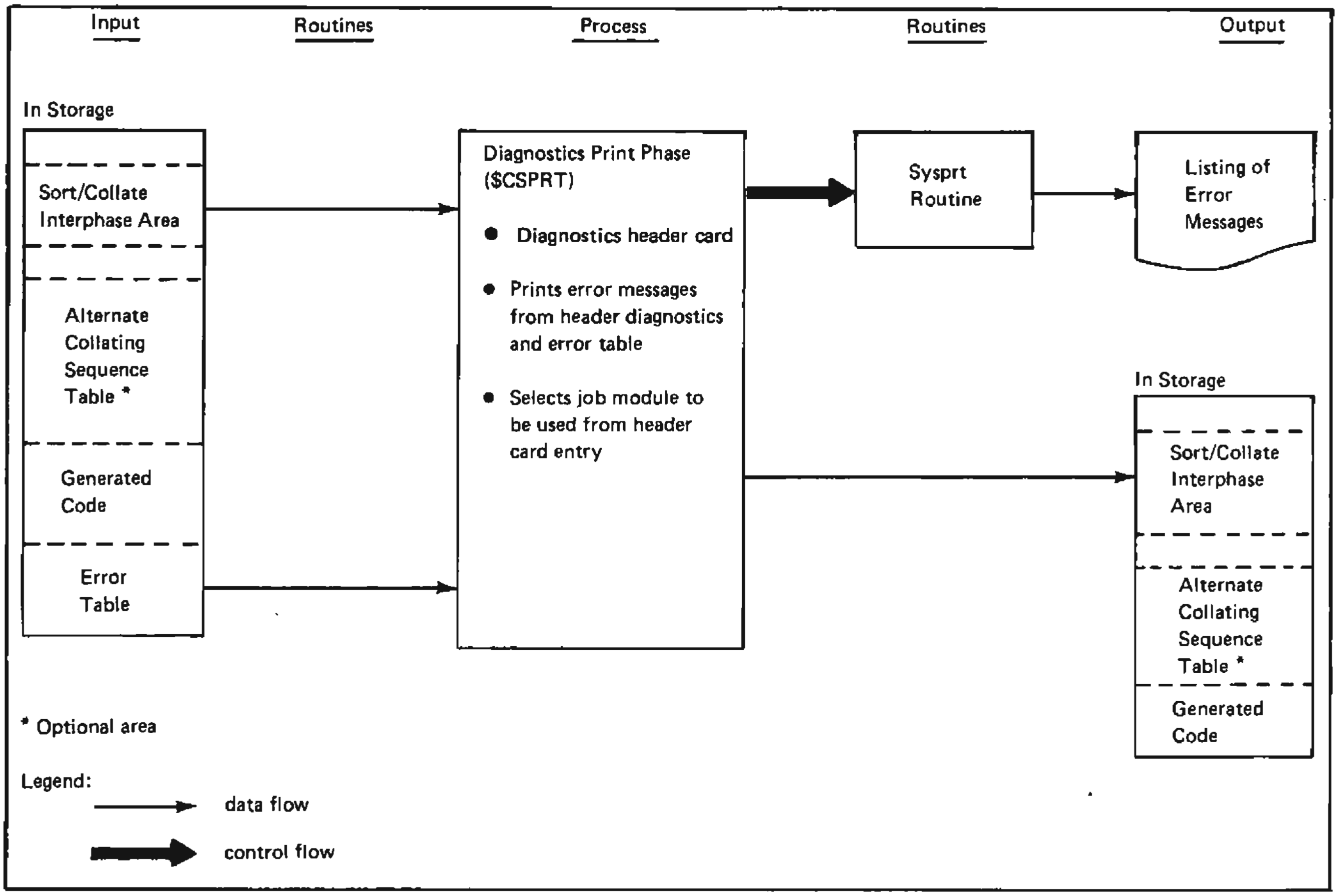
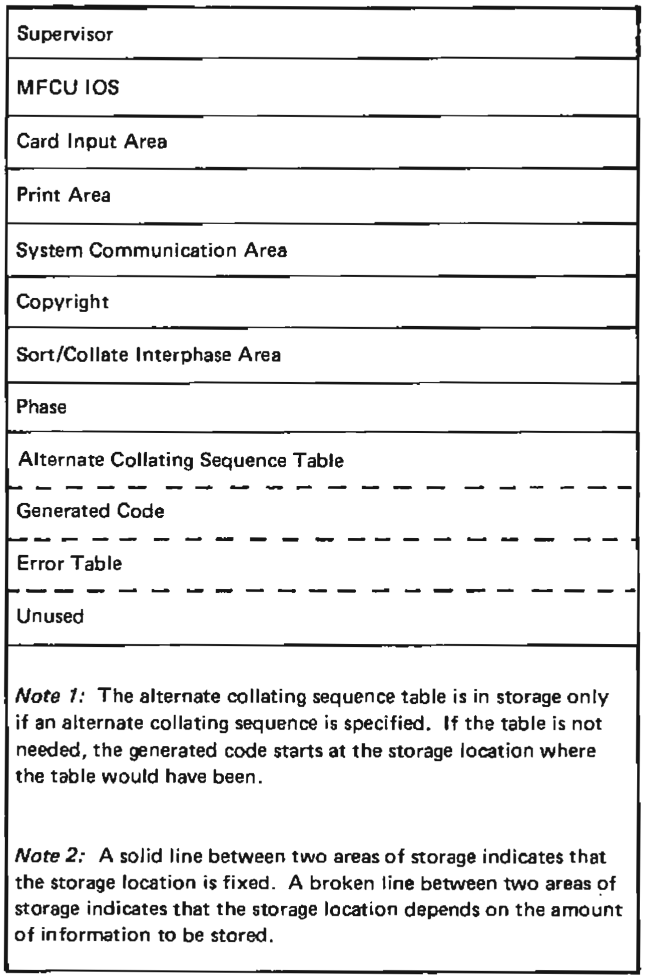

Input
Routines
Process
Routines
Output
In Storage
Sort/Collate Interphase Area
Alternate Collating Sequence Table *
Generated Code
Error Table
Diagnostics Print Phase ($CSPRT)
● Diagnostics header card
● Prints error messages from header diagnostics and error table
● Selects job module to be used from header card entry
Sysprt Routine
Listing of Error Messages
In Storage
Sort/Collate Interphase Area
Alternate Collating Sequence Table *
Generated Code
* Optional area
Legend:
data flow
control flow

Figure 3-2. Functional Flow of Data and Control for Sort/Collate Diagnostics Print Phase ($CSPRT)

Section 3. Program Organization

This section gives a detailed description of the Generation and the Diagnostic Print phases. Each major function is explained individually, and its entry point to the program is given. The entry point, exit point, input and output for the phase are also listed. For a description of the Full Function MFCU IOS routine used, refer to the *IBM System/3 Disk System Data Management and Input/Output Supervisor Logic Manual*, SY21-0502. Figure 3-3 shows a storage map for the phases.



Note 1: The alternate collating sequence table is in storage only if an alternate collating sequence is specified. If the table is not needed, the generated code starts at the storage location where the table would have been.

Note 2: A solid line between two areas of storage indicates that the storage location is fixed. A broken line between two areas of storage indicates that the storage location depends on the amount of information to be stored.

Figure 3-3. Storage Map of Generation and Diagnostic Print Phases

Generation Phase ($CSORT)

Entry Point: ASMAA1 from the supervisor.

Functions:

- Does initialization (ASMAA1). This section of the program is overlaid when code is generated.

  1. Checks for copyright violation.

  2. Reads first card.

- Processes header card (AAB100). This section of the program is overlaid when code is generated.

  1. Checks for the header control card; halts if not found.

  2. Checks the print option; suppresses printing if specified.

  3. Initializes the print area to blanks.

  4. Prints the Sort/Collate heading line.

  5. Prints the header card image on the system printer.

  6. Saves the header card entries in the Sort/Collate interphase area.

  7. Checks for alternate collating sequence.

- Builds the alternate collating sequence table if an alternate collating sequence specified (AAN100). This section of the program is overlaid when code is generated.

  1. Reads the alternate collating sequence cards (ALTSEQ).

  2. Prints ALTSEQ card images on the system printer.

  3. Diagnoses the ALTSEQ control cards for valid hexadecimal entries.

  4. Modifies the alternate collating sequence table according to the entries found on the ALTSEQ control card.

- Prints specifications and does end-of-file processing (AAA100).

  1. Reads specification cards.

  2. Prints source listing.

  3. End-of-file processing:

     a. Moves last code segment to the generated code area.

     b. Stores the address of the last entry of the error table in the Sort/Collate interphase area.

- Identifies include, omit, and field cards (AAC100).

  1. Determines card type.

  2. Checks order of specification cards.

- Processes include and omit cards (AAD100).

  1. Diagnoses specification errors.

  2. Generates proper code segments.

- Processes field cards (AAG100).

  1. Diagnoses specification errors.

  2. Generates proper code segments.

- Moves generated code segments to the generated code area unless a terminal error has been found (AAH100).

- Determines zone and fills in part of zone test code segment (AA1100).

- Calculates lengths and displacements (AAJ100).

  1. Uses Factor 1 data on include or omit cards to calculate read area displacement and field length.

  2. Uses Factor 2 contents on include or omit cards to calculate work area displacement.

  3. Uses Location field data on field cards to calculate read area displacement and field length.

- Converts decimal numbers to binary (AAK100).

  1. Checks for valid entry (01-96). If an error is found, a value of 01 is assumed.

  2. Converts Sum of Lengths of Control Fields entry in header card.

  3. Converts contents of Factor 1, Factor 2, and Location fields.

- Builds error table as errors are found (AAL100).

- Translates constants entered on the include or omit cards to the collating sequence if an alternate collating sequence is specified (AAM100).

- Processes comment cards (AAP100).

  1. Checks for a comment card (an * in column 7).

  2. Prints comment card.

  3. Reads next card if it is a comment card.

Exit: To the supervisor to call the Diagnostics Print phase.

Input:

- Header cards.
- Alternate collating sequence cards.
- Record Type cards.
- Control Field cards.

Output:

- Generated code in storage.
- Error table, in storage, of all errors found on the specification cards and a source listing of all specification cards read (if the logging device has been turned on by the / / LOG Operation Control Language statement).
- Alternating collating sequence table in storage if an alternate collating sequence is specified.
- Sort/Collate interphase area information.

Routines Called:

- Full Function MFCU IOS.
- Halt/Syslog.

Diagnostics Print Phase ($CSPRT)

Entry Point: ASMAB1 from the Supervisor.

Functions:

- Diagnoses header card (ABB100).
- Selects job module to be used from header card entry (ABB100).
- Scans error table for errors (ABC100).
- Unpacks statement numbers (ABD100).
- Prints listing of error numbers and messages from header diagnostics and error table (ABE100) if the logging device has been turned on.

Exit: To the supervisor to call the requested job module.

Input:

- Header information in the Sort/Collate interphase area.
- Error table in storage.

Output: Listing of the statement number of the card in error and its error message (if the logging device has been turned on).

Routines Called: Halt/Syslog.

Section 4. Data Area Formats

Copyright

This 46-byte area contains the program number for this program and copyright information as follows: 5702- UT1ƀCOPYRIGHTƀIBMƀCORPƀ1970 (The ƀ represents a blank). The remainder of the area is filled with blanks. This area remains in storage throughout all phases.

Sort/Collate Interphase Area

This 34-byte Sort/Collate interphase area is established by the Generation phase and is used to store information that will be needed by the Generation, Diagnostics Print and Execution phases. Figure 3–4 shows the format and contents of the Sort/Collate interphase area. See the Header section of the Sequence Specifications sheet for the columns referred to by this figure.

		Phase		
Bytes	Definition	Generation	Diagnostics Print	Execution
1	Sum of control field lengths (col 13-17)	I, DG, M		R
2	Type of sequencing as specified (col 18)	I, R	DG	R
3	SEC unmatched stacker as specified (col 19)	I	DG	M (match job only)
4	SEC matched stacker as specified (col 20)	I	DG	M (match job only)
5	PRI matched stacker as specified (col 21)	I	DG	M (match job only)
6	PRI unmatched stackers as specified (col 22)	I	DG	M (match job only)
7	SEC omit stacker as specified (col 23)	I	DG	M (match job only)
8	PRI omit stacker as specified (col 24)	I	DG	M (match job only)
9	Number (col 25); 1=X'F1' N=X'C5'	I	DG	R (match job only)
10-13	Branch instruction	D	R	M (generated code uses to return to job module)
14-17	Branch to entry of generated code	D, M (if ALTSEQ)		R (job module uses to enter generated code)
18-19	Address of Input Area	D		R
20-21	Address last byte of spec hold area	D	M (to CWA)	M (to CWA+sum lengths) M (to IN+95, if ALTSEQ)
22	Bits—Interphase Switches			
	0 0/1=Current record is an Include/Omit	D		M (for each input record)
	1 1 =Have Omit Records	D		R
	2 1 =Select job type	D		R
	3 1 =Suppress Print during current phase	D, R	M	R
	4-7 1111=Sort; 0111=Merge; 0011=Match; 0001=Select	D	M	R

Figure 3-4. Sort/Collate Interphase Area Format (Part 1 of 2)

		Phase		
Bytes	Definition	Generation	Diagnostic Print	Execution
23	Bits—Stacker Information (Select Job Only)			M (for each input record)
	6-7 01, 10, 11=Stackers 1, 2, 3	D		
24-29	Job type as specified (col 7-12)	I	DG	
30-31	Address of error table (last storage location)	D	R	
32-33	Address of last error in table	D	M	
34	Bits—Generation—Diagnostic Switches			
	0 1=Control Field (F) cards present	D, M	R	
	1 1=Suppress printing during execution	D, M D, M	R	
	2 1=Type assumed			
	3 1=T type error (terminate)	D, M	M	
	4 1=W type error (warning)	D, M	R	
	5 1=First card in set	D, M		
	6 1=Include—All present	D, M		
	7 1=Alternate Collating Sequence	D, M	R	

I=input from S/C Header card
D=defined by this phase
DG=diagnosed by phase
M=modified by this phase
R=referenced by this phase (not modified)

Figure 3-4. Sort/Collate Interphase Area Format (Part 2 of 2)

Section 5. Object Program

The object program is made up of two parts: the selected job module and the generated code. The selected job module reads a record and branches to the generated code. The generated code identifies the record and builds the control word; then it branches back to the job module. The job module finishes processing the record and controls its stacker selection. Figure 3-5 shows the control flow of the object program.

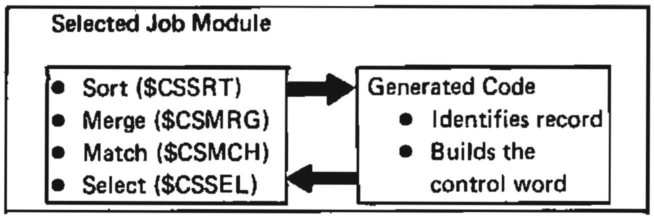
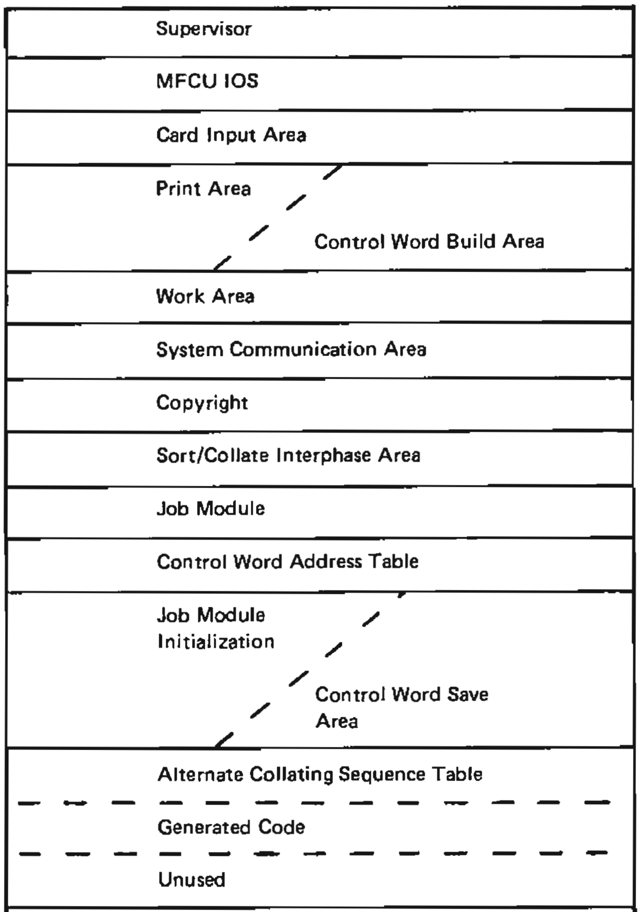


Figure 3-5. Control Flow of Object Program

JOB MODULES

The Diagnostics Print phase sets switches in the Sort/Collate interphase area to identify the job type specified on the header card. At the beginning of the Execution phase, the Supervisor loads the job module specified on the header card into storage. Figure 3-6 shows a map of storage during execution of one of the job modules.


Supervisor
MFCU IOS
Card Input Area
Print Area
Control Word Build Area
Work Area
System Communication Area
Copyright
Sort/Collate Interphase Area
Job Module
Control Word Address Table
Job Module Initialization
Control Word Save Area
Alternate Collating Sequence Table
Generated Code
Unused

Note 1: The diagonal broken line indicates that the two area divided by this line overlay each other whenever necessary.

Note 2: The alternate collating sequence table is in storage only if an alternate collating sequence is specified. If the table is not needed, the generated code starts at the storage location where the table would have been.

Note 3: A solid line between two areas of storage indicates that the storage location is fixed. A broken line between two areas of storage indicates that the storage location depends on the amount of information to be stored.

Figure 3-6. Storage Map of Execution Phase

Descriptions and flowcharts for each of the four job modules follow. Each job module is made up of several routines. The main flowchart for each module is shown first (Charts FA–FD), followed by flowcharts of routines which are branched to from various parts of the modules (Charts FE–FG).

Sort Job Module ($CSSRT)

This job module arranges a deck of cards into a specified order (either ascending or descending). If omit records are specified or implied, only stackers 1 and 3 are used for sorting during the first pass. The omit records are separated from the rest of the deck and selected to stackers 2 and 4. A program halt occurs if any omit records are re-entered in later passes.

During all other passes, all four stackers are used. Records are merged from the two hoppers to establish sort strings. For an ascending sort (descending sort is opposite), stacker selection is as follows:

1. The record from the primary or secondary hopper with the lowest control word value is chosen to be processed.

2. The stacker whose last record has a lower or equal control word value is selected. If the last record in each of two or more stackers is lower than the current control word, the stacker with the minimum difference between the two control word values is selected.

3. If none of the stackers can be selected, step 1 is repeated, but this time the record from the other hopper with higher control word value, is chosen, provided that hopper is not empty.

4. Step 2 is repeated for the higher control word value.

5. If none of the stackers can be selected, the first record is forced to the stacker with the highest control word value.

6. Stacker selection continues until both hoppers reach end of file.

In order to make the correct stacker selection, this job module uses the control word address table (see *Data Areas, Control Word Address Table* in this section for the contents). First the module searches through the table for a stacker control word lower than the current hopper control word. It chooses the stacker control word closest to, but less than, the current hopper control word. If the search is successful, the stacker control word is replaced by the current hopper control word, and a new card is read from the current hopper.

If the search is not successful, a new string is started by placing the lower (or only) hopper control word in the highest stacker. The control word address table is then shifted by saving the new low stacker address and shifting the table to the right three bytes. The new low stacker control word address is then placed in the table as the lowest stacker.

Continuous passes are made through the card deck until one sequenced string is produced. A halt occurs at the end of each pass and at end of job. Chart FA is a flowchart of this job module.

Merge Job Module ($CSMRG)

This job module is a one-pass run which collates a sequenced file from the primary hopper with a similarly sequenced file from the secondary hopper to create one merged file.

First the record read is sequenced checked. If it is not in order, a program halt occurs. Next, the records are merged. For an ascending merge (descending merge is opposite) the control word of a primary record is compared to the control word of a secondary record. The smaller control word is selected, and that record is merged. If the control word of a primary record is equal to the control word of a secondary record, the primary record is merged before the secondary record. Merged cards are selected to stacker 1.

Records to be omitted during the merge run are selected to stacker 2 if they were in the primary hopper and to stacker 4 if they were in the secondary hopper. Undefined records are included with omitted records. Chart FB is a flowchart of this job module.

Match Job Module ($CSMCH)

This job module compares two card files in the same sequence to find the records that match. There are two types of matching: 1 for 1 and N for N. If column 25 on the header card contains a 1, one primary record can be matched with one secondary record. If column 25 contains an N, multiple primary records can be matched with multiple secondary records.

First, the record read is sequenced checked. If it is not in order, a program halt occurs. The control word of the primary record is then compared to the control word of the secondary record. If the two control words match, the records are selected to the stackers designated as the primary matched and secondary matched stackers. Primary records are selected to the stacker before secondary records. If the control words do not match, the record having the low control word (high control word if descending order is used) is selected to the stacker designated for its unmatched records.

Records to be omitted are selected to the stackers designated as the primary omit and secondary omit stackers. Chart FC is flowchart of this job module.

Select Job Module ($CSSEL)

This job module selects specified records from a file and puts them in the specified stacker (stacker 1, 2, or 3). The rest of the file is left in its original order and put in stacker 4. If a sequence has been specified by the user, each selected record is sequence checked. A program halt occurs if the records are not in order. Chart FD is a flowchart of this job module.

CODE SEGMENTS

The generated code is built in storage during the Generation phase. These code segments reflect the specifications on the Sequence Specifications sheet. They are placed in storage in the order in which they are generated.

When the job module being used during the Execution phase reaches the record identification section, it branches to the generated code. The two main functions of the generated code are to identify the record and build the control word.

The record identification logic is comprised of one or more sets. A set is made up of the first record specification (either an include or omit record) up to, but not including, the first specification of another type. Each set begins with an IYES branch.

Each set is made up of subsets. A subset is the part of a set which meets one of the following requirements:

1. Beginning of the set up to the first OR following an AND specification.

2. Beginning of the following subset to the first OR following an AND specification.

3. Beginning of the following subset to the end of the set.

4. Beginning of the set to the end of the set.

Each subset begins with an INO branch and ends with an unconditional branch to IYES. The instructions between the two branch statements in the subset consist of the code segments for identifying the record. Figure 3-7 shows the general structure of a set and its subsets.

If a card is identified as an include record, the control fields of this record are then assembled to create the control word (omit records do not have control words). The length of the control word is specified by the Sum of Lengths of Control Fields entry on the header card. The control word is built from left to right in the order in which the control fields are specified on the Sequence Specifications sheet (Figure 3-8). If the fields for the record are less then the total length, the control word is padded to the right with hexadecimal zeros.

When a record has been identified and the specified control word built, a switch is set to indicate whether the card is an include or an omit record. The generated code then returns to the job module.

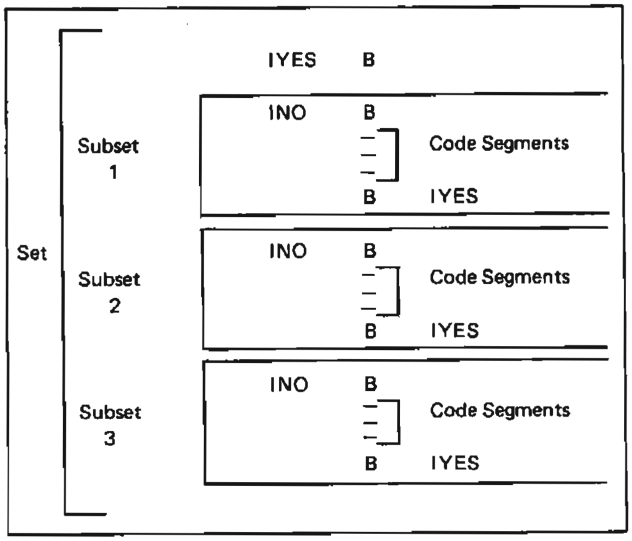


Figure 3-7. Set and Subset Structure

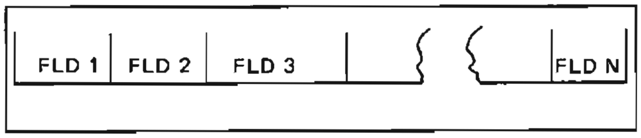


Figure 3-8. Building the Control Word

Chart FH shows a flowchart of the record identification routine and the code segments which are related to a particular part of the routine. A description of each of these code segments follows. The following abbreviations are used in the instructions which make up the code segments:

Abbreviation	Meaning
D1	To Location (columns 13–16) minus 1 of Factor 1 (columns 9–16) or To Location (columns 13–16) of Control Field card.
D2	To Location (columns 24–27) minus 1 of Factor 2 (columns 20–39).
L	Length of Factor 1 (columns 9–16) or of the Location (columns 9–16) field.
CONST	Address of the Factor 2 (columns 20–39) constant.
WKA	Work area = X'DF' + L.

CWA	Control word area = X'7B' + sum of the lengths of the control fields. This area points to the right end of the field entry within the control word. CWA is a variable dependent upon the fields specified within a particular record. If, for example, three fields make up a control word for a particular record (Figure 3-8), CWA would be:
	Length of Field — CWA
	FLD1=10 — X'7B'+X'0A'
	FLD2=3 — X'7B'+X'0D'
	FLD3=6 — X'7B+X'13'

Branch to Job Module

```
INCLUD SBF          SWITCH, SWMK

       B            JOBMOD

OMIT   SBN          SWITCH, SWMK

       B            JOBMOD
```

These instructions are always present at the beginning of the generated code. After the code segments identify the record and build the specified control word, the Branch to Include/Omit code segment branches to either INCLUD or OMIT depending on the record type. These instructions set a switch to indicate the record type and return to the job module.

Record Identification Code Segments

Beginning of a Set

```
IYES         B            BCW/OMIT
```

The code segments, which identify the record type, branch to this code segment if the record does meet the specifications. The branch IYES takes depends on the record type. For an omit record, IYES branches to the instruction which sets a switch informing the job module that the current record should be omitted (OMIT). For an include record, IYES branches to the code segments which build the control word (BCW).

Beginning of a Subset

```
INO          B            NXTSET
```

The code segments which identify the record type branch to this code segment if the record does not meet the specifications. INO then branches to the first instruction following the INO branch of the next subset including the first subset of a new set (NXTSET). The last subset is an implied omit. It branches to the instruction which sets a switch informing the job module that the current record should be omitted.

Set Stacker for Stacker Select

```
MVI        STSLK1, X'01'       Moves specified
                               stacker number
```

This code segment moves the stacker specified in column 9 to the Sort/Collate interphase area when an include card is being identified during a select job. X'01' is a default number so stacker 1 will be used if no stacker is specified.

Jump Over Constant

```
           J        CONST+1
CONST      DC       CL1'constant'
```

Constants are placed in the program as they are encountered in the specifications. This code segment jumps the length of the constant to continue with the rest of the program.

Character -- Field to Constant

```
CLC        D1(L,XR1), CONST
```

If a C is specified in column 8 and a C is specified in column 19, this code segment tests the relationship of the characters in the positions specified by Factor 1 to the constant specified.

Character -- Field to Field

```
CLC        D1 (L,XR1),D2(,XR1)
```

If a C is specified in column 8 and an F is specified in column 19, this code segment tests the relationship of the characters in the positions specified by Facter1 to the characters in the positions specified by Factor 2.

Zone

```
        MNN  COMP+1,D1(,XR1)        Set numeric equal
COMP    CLI  D1(,XR1),X'Z0'         Are zones equal
        BE   IYES/INO/*+11
        CLI  D1(,XR1),C' '          Special zone test
```

If a Z is specified in column 8, this code segment tests the relationship of the zone portion of the positions specified by Factor 1 to the zone portion of the constant specified.

The zone test for a C, D, and F zone or an &, minus (-), and a blank character uses the entire code segment. The branch that the BE instruction takes is determined as follows:

1. EQ relationship specified

Current Card (col 7)	*Next Card (col 7)*	*Branch To*
A/ƀ/O	A	*+11
A/ƀ/O	ƀ/O	IYES

2. NE relationship specified

Current Card (col 7)	*Next Card (col 7)*	*Branch To*
A/ƀ/O	A	INO
A	ƀ/O	INO
ƀ/O	ƀ/O	*+11

All other zone tests use only the first two instructions of the code segment because a test for special characters is not necessary.

Digit -- Field to Constant

```
ZAZ    WKA(L,XR1),D1(L,XR1)     Move Factor 1,
                                clearing zones
SBN    WKA(,XR1),X'F0'          Set last zone
                                positive
CLC    WKA(L,XR1),CONST
```

If a D is specified in column 8 and a C is specified in column 19, this code segment tests the relationship of the digit portion of the positions specified by Factor 1 to the digit portion of the constant specified.

Digit -- Field to Field

```
ZAZ    WKA(L,XR1),D2(L,XR1)       Move Factor 2,
                                  clearing zones
SBN    WKA(,XR1),X'F0'            Set last zone
                                  positive
ZAZ    WKA+L(L,XR1),D1(L,XR1)     Move Factor 1,
                                  clearing zones
SBN    WKA+L(,XR1),X'F0'          Set last zone
                                  positive
CLC    WKA+L(L,XR1),WKA(,XR1)
```

If a D is specified in column 8 and an F is specified in column 19, this code segment tests the relationship of the digit portion of the positions specified by Factor 1 to the digit portion of the positions specified by Factor 2.

Unpacked—Field to Constant

```
ZAZ    WKA(L,XR1),D1(L,XR1)     Move field to work
                                area, setting sign

SZ     WKA(L,XR1),CONST(L)      Set condition code
```

If a U is specified in column 8 and a C is specified in column 19, this code segment tests the relationship of the signed decimal field in the positions specified by Factor 1 to the constant specified.

Unpacked—Field to Field

```
ZAZ    WKA(L,XR1),D2(L,XR1)         Move Factor 2
                                    setting sign

ZAZ    WKA+L(L,XR1),D1(L,XR1)       Move Factor 1
                                    field to work
                                    area

SZ     WKA+L(L,XR1),WKA(L,XR1)      Set condition
                                    code
```

If a U is specified in column 8 and an F is specified in column 19, this code segment tests the algebraic relationship of the unpacked decimal fields specified by Factor 1 and Factor 2.

```
BC      IYES/INO
```

This code segment follows each record identification test and is used if a particular test is met. The exact instruction depends on the record specifications. Figure 3-9 shows the resulting branch instruction and where it will branch. For example, if an O is specified in column 7 of the current specification card, and an O is specified in column 7 and an NE relationship is specified in columns 17-18 of the current specification card, a BNE to IYES is generated.

Relationship Specified (Col 17-18)	Current Card	ƀ/O	ƀ/O	A	A/ƀ/O
	Next Card	ƀ/O	A	A/ƀ/O	No more cards for this record type
EQ		BE to IYES	BNE to INO		
NE		BNE to IYES	BE to INO		
LT		BL to IYES	BNL to INO		
GT		BH to IYES	BNH to INO		
LE		BNH to IYES	BH to INO		
GE		BNL to IYES	BL to INO		

Note: The entries given for Current Card and Next Card refer to the entries for column 7 of the current specifications and the next specifications as follows:

ƀ = blank
O = OR
A = AND

Figure 3-9. Table of Branch on Condition Instructions

Branch Instruction

```
B         IYES/INO
```

This code segment occurs at the end of a subset and is used if none of the tests in the subset are met. The branch taken is dependent on the previous Branch on Condition instruction. If the Branch on Condition instruction branches to IYES, the Branch instruction branches to INO; likewise, if the Branch on Condition instruction branches to INO, the Branch instruction branches to IYES.

Control Word Code Segments (Control Field Card)

Jump Over Constant

See the *Jump Over Constant* code segment under *Record Identification Code Segments* in this section.

Normal Field – Character

```
MVC     CWA(L,XR1),D1(,XR1)
```

If an N is specified in column 7 and a C is specified in column 8, this code segment moves the characters of the positions specified by the Location field to the control word build area.

Normal Field – Zone

```
MZZ     CWA(,XR1),D1(,XR1)
SBF     CWA(,XR1),X'0F'          Set numeric
                                 portion off
```

If an N is specified in column 7 and a Z is specified in column 8, this code segment moves the zone portion of the position specified by the Location field to the control word build area.

Normal Field -- Digit

```
ZAZ     CWA(L,XR1),D1(L,XR1)     Move Factor 1,
                                 clearing zones
SBN     CWA(,XR1),X'F0'          Set last zone
                                 positive
```

If an N is specified in column 7 and a D is specified in column 8, this code segment moves the digit portion of the position specified by the Location field to the control word build area.

Opposite Field -- Digit

```
MVI     CWA(,XR1),X'F9'
MVC     CWA-1(L-1,XR1),CWA(,XR1)
MZZ     CWA(,XR1),D1(,XR1)
SZ      CWA(L,XR1),D1(L,XR1)
SBN     CWA(,XR1),X'F0'
```

If an O is specified in column 7, and D is specified in column 8, this code segment moves 9's into the control word area with the same sign as the Location fields. The digit portion of the information specified in the Location field is then subtracted from the 9's so that the opposite digit remains in the control word build area.

Unpacked Field—Normal or Opposite

```
ZAZ     CWA(L,XR1),D1(L,XR1)     Move field
                                 to control
                                 word

JC      13,HIGH/LOW              Condition is
                                 low for
                                 opposite, high
                                 for normal

MVC     WKA+15(16,XR1),FFCON     Move X'FF's
                                 to work area

SLC     WKA(L,XR1),CWA(,XR1)     Complement

MVC     CWA(L,XR1),WKA(,XR1)     Back to
                                 control word
```

If a U is specified in column 8, an unpacked field is assembled. The sign of the unpacked number determines whether the number is complemented by subtracting it from a field of X'FF's.

Force Sequence -- Leading Instruction

```
MVI     CWA(,XR1),X'FF'
```

If an F is specified in column 7, and the first force line does not have a continuation punch, this code segment is generated as a forced sequence test is entered. It moves the highest possible value into the control word build area for an ascending sequence (X'FF') and the lowest possible value for a descending sequence (X'OO'). If none of the record characters are found, this default value is used in the control word.

Beginning of Force Lines

```
         J     4
BGFORC   B     ENFORC
```

This code segment is used to leave the forced sequence tests. ENFORC is the first instruction past the current series of forced sequence tests.

Forced Field -- Character (Part 1)

```
CLI      D1(,XR1),X'00'        Is character of
                               input equal
```

If an F is specified in column 7 and a C is specified in column 8, this code segment compares the character of the position specified by the Location field to the record character specified in column 17.

Forced Field -- Zone (Part 1)

```
        MNN  COMP+1,D1(,XR1)    Set numeric equal
COMP    CLI  D1(,XR1),X'Z0'     Are zones equal
        BE   *+10
        CLI  D1(,XR1),C'        Special zone test
```

If an F is specified in column 7 and a Z is specified in column 8, this code segment tests the zone portion of the position specified by the Location field to the zone portion of the record character specified in column 17.

The zone test for a C, D, and F zone or an &, minus (-), and a blank character uses the entire code segment. For all other zone tests, only the first two instructions are used because a test for special characters is not necessary.

Forced Field -- Digit (Part 1)

```
          MZZ  COMP+1,D1(,XR1)    Set zone equal
COMP      CLI  D1(,XR1),X'0D'     Are digits equal
```

If an F is specified in column 7 and a D is specified in column 8, this code segment compares the digit portion of the position specified by the Location field to the digit portion of the record character specified in column 17.

Force-All

```
MVI      CWA(,XR1),X'00'        Unconditional
                                move of substi-
                                tute character
B        BGFORC
```

If a force-all line is indicated, this code segment moves the Substitute Characters specified in column 18 to the control word build area without any testing.

Forced Field -- Character, Zone, Digit (Part 2)

```
JNE      7                      Doesn't meet test,
                                try next
MVI      CWA(,XR1),X'00'        Meets test, move
                                in substitute
                                character
B        BGFORC
```

If the contents of specified position (digit portion, zone portion, or character of a forced field) does not compare equal, a jump is taken, and the next code segment is executed. If the information does compare equal, the substitute character specified in column 18 is moved to the control word build area. A branch is then taken to leave the forced sequence tests.

Branch to Include/Omit

```
B        INCLUD/OMIT
```

This code segment branches to the *Branch to Job Module* at either of two times:

- After the control word is built
- Immediately after the record is identified, if no control fields are specified.

DATA AREAS

Sort/Collate Interphase Area

See *Section 4, Data Area Formats,* for information on this data area.

Control Word Address Table

This 12-byte table is used by the Sort job module to establish sort strings. If omit records are specified, only half of the table is used on the first pass. Figure 3-10 shows the format and original contents of the table.

The order of the entries in the table changes because the addresses of the stacker control words (control word of the last card in the stacker) are placed in the table in ascending order according to the values of the stacker control words. Therefore, the address of the lowest stacker control word is first. In a descending sequence, the address of the highest stacker control word is first.

Displacement		3 bytes	
Dec	Hex		
+0	0	Stacker select bits for stacker 4 (X'04')	Address of stacker 4 control word
+3	3	Stacker select bits for stacker 2 (X'06')	Address of stacker 2 control word
+6	6	Stacker select bits for stacker 3 (X'07')	Address of stacker 3 control word
+9	9	Stacker select bits for stacker 1 (X'05')	Address of stacker 1 control word
+12	C		

Figure 3-10. Control Word Address Table

SAMPLE DUMP ANALYSIS

This section is presented as an aid for examining the areas of a storage dump of a Sort/Collate program. Figure 3-11 shows the source listing of the specification cards. Figure 3-12 shows a sample storage dump for the program.

Figure 3-13 shows a symbolic representation of the code generated for this program.

The different areas in storage and their locations are as follows:

Full Function MFCU (Compiler) IOS: 0000

Card Input Area: 0F00

Print Area/Control Word Build Area: 0F7C
Before any printing is done, this area is filled with blanks; before a control word is built, this area is filled with hexadecimal zeros.

Work Area: 0FE0

System Communication Area: 1000

Copyright: 10B2

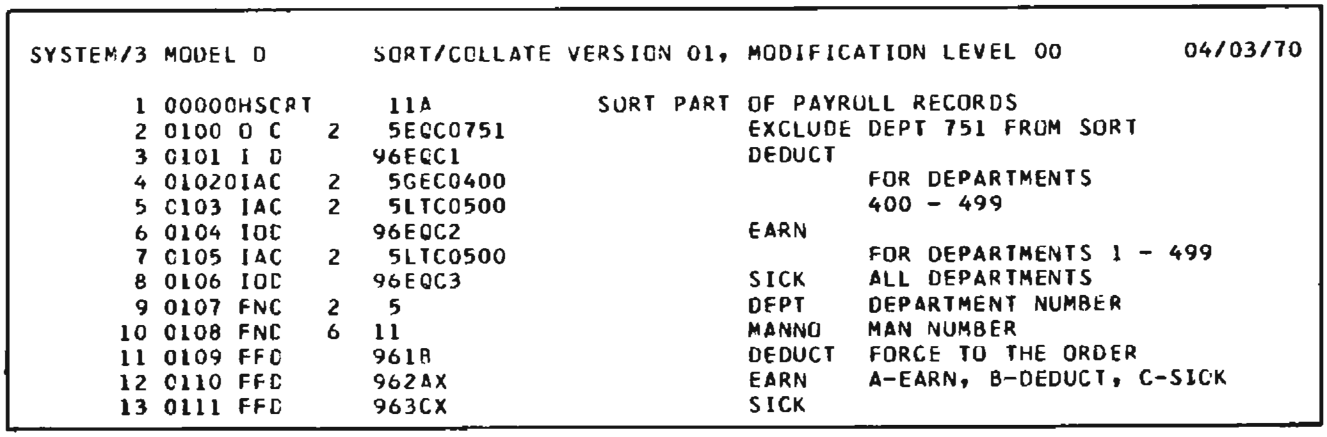
Sort/Collate Interphase Area: 10E0

Phase or Job Module: 1106

Alternate Collating Sequence Table: 1F20

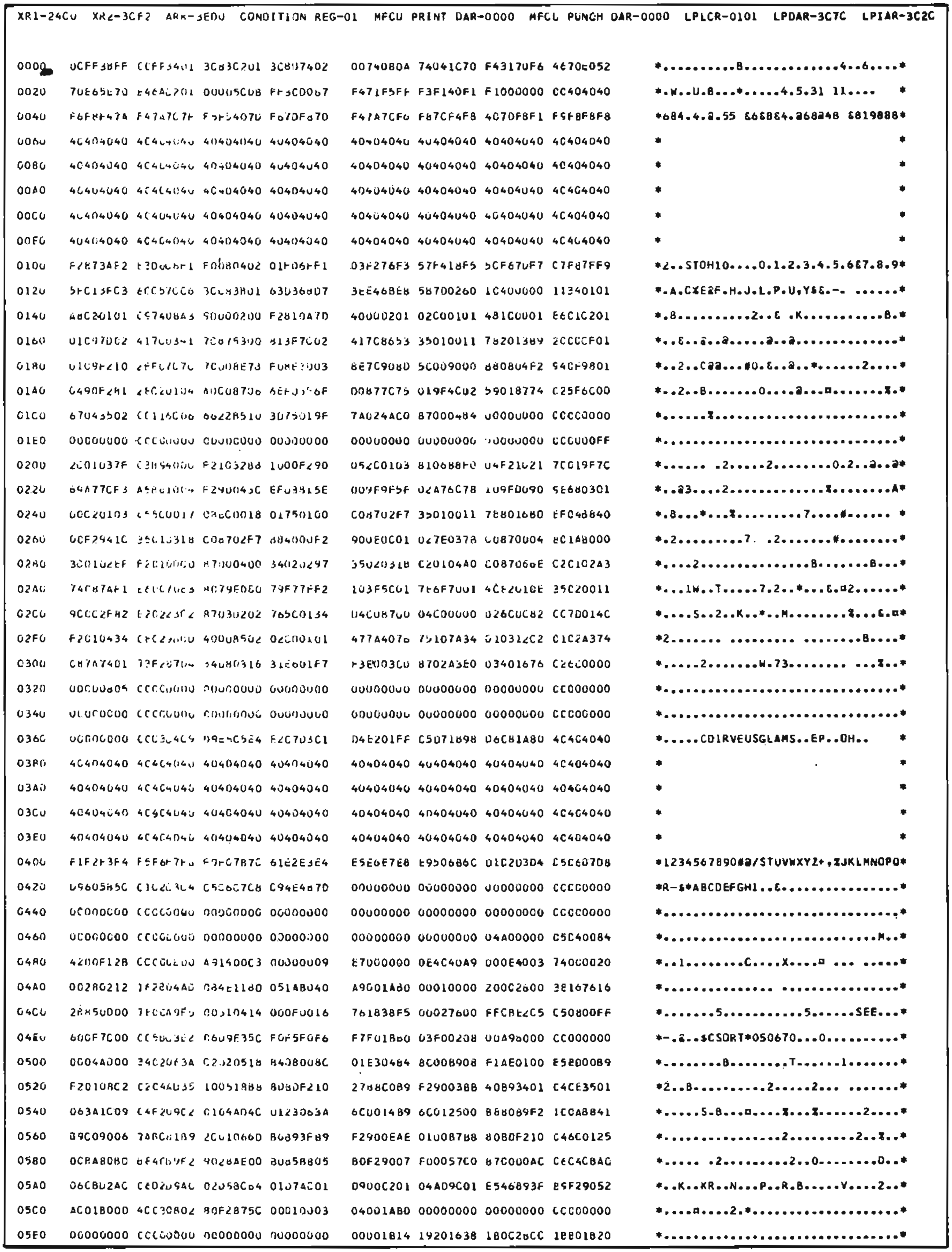
Generated Code: Branch is found in bytes 14-17 of the Sort/Collate interphase area.

Error Table: Location depends on the storage size used (storage size is found in the system communication area.)



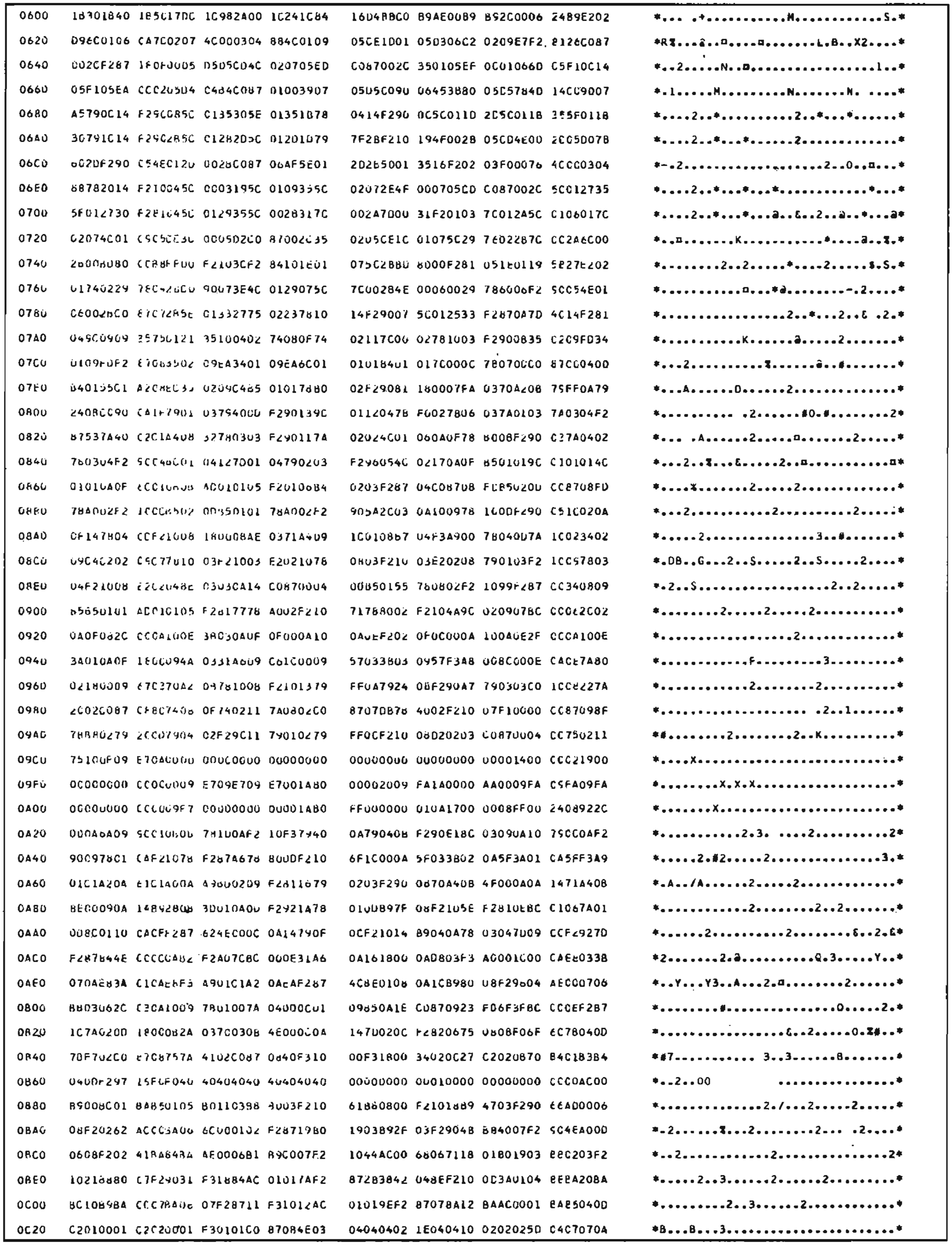
```
SYSTEM/3 MODEL D       SORT/COLLATE VERSION 01, MODIFICATION LEVEL 00       04/03/70

         1 00000HSCRT      11A          SURT PART OF PAYRULL RECORDS
         2 0100 O C   2     5EQC0751              EXCLUDE DEPT 751 FROM SORT
         3 0101 I D        96EQC1                 DEDUCT
         4 01020IAC   2     5GEC0400                      FOR DEPARTMENTS
         5 0103 IAC   2     5LTC0500                      400 - 499
         6 0104 IOD        96EQC2                 EARN
         7 0105 IAC   2     5LTC0500                      FOR DEPARTMENTS 1 - 499
         8 0106 IOD        96EQC3                 SICK    ALL DEPARTMENTS
         9 0107 FNC   2     5                     DEPT    DEPARTMENT NUMBER
        10 0108 FNC   6    11                     MANNO   MAN NUMBER
        11 0109 FFD        961B                   DEDUCT  FORCE TO THE ORDER
        12 0110 FFD        962AX                  EARN    A-EARN, B-DEDUCT, C-SICK
        13 0111 FFD        963CX                  SICK
```

Figure 3-11. Source Listing of Specification Cards



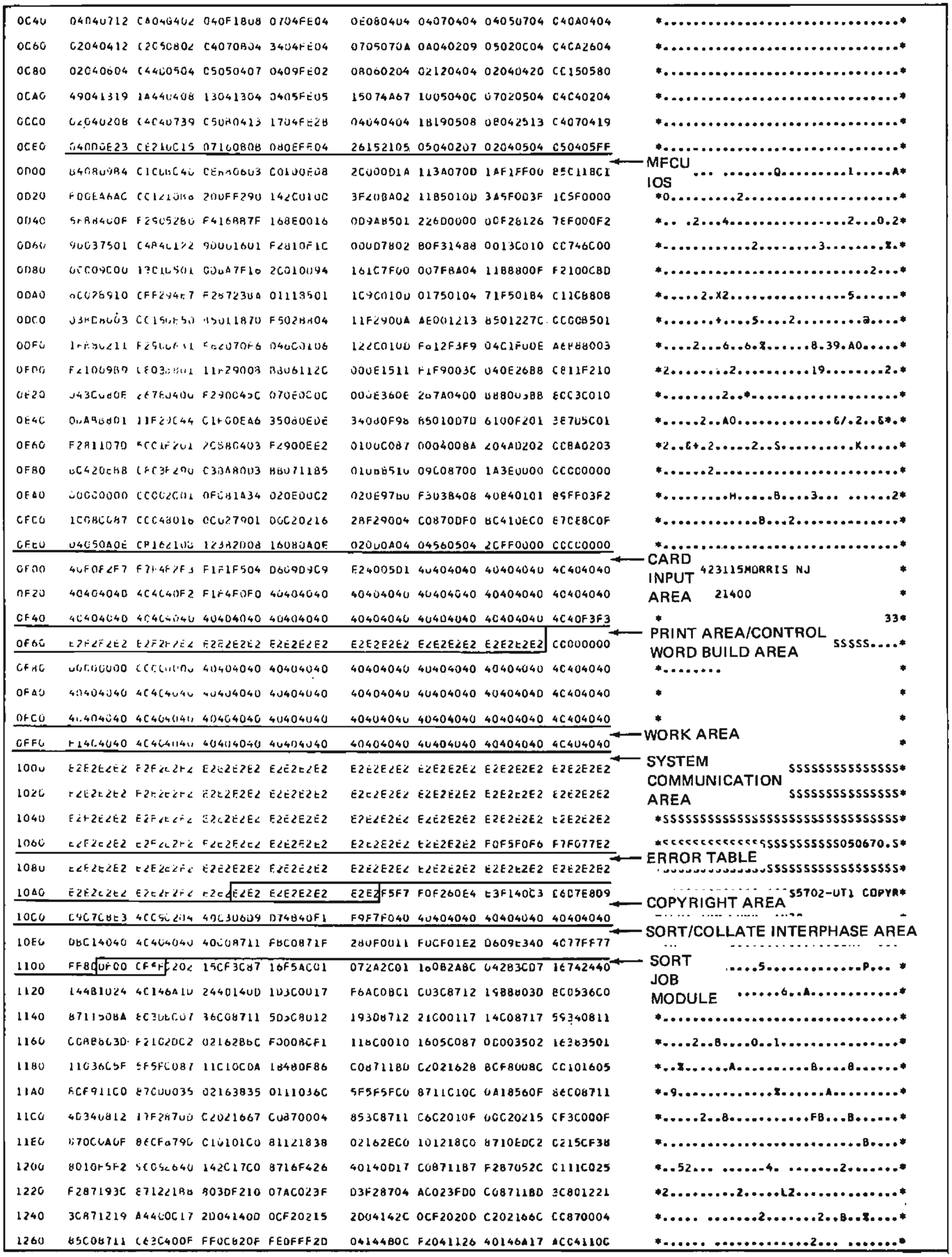
```
XR1-24C0  XR2-3CF2  ARR-3E00  CONDITION REG-01  MFCU PRINT DAR-0000  MFCU PUNCH DAR-0000  LPLCR-0101  LPDAR-3C7C  LPIAR-3C2C


0000   0CFF3BFF CCFF3401 3C83C201 3C807402   0074080A 74041C70 F43170F6 4670E052   *..........B.............4..6....*
0020   70E65E70 E46AC701 00005C08 FF3C0067   F471F5FF F3F140F1 F1000000 CC404040   *.W..U.8...*.....4.5.31 11....   *
0040   F6F8F47A F47A7C7F F5F5407D F67DF87D   F47A7CF6 F87CF4F8 407DF8F1 F9F8F8F8   *684.4.@.55 &6&884.@68@48 &819888*
0060   4C404040 4C4C4040 40404040 40404040   40404040 40404040 40404040 40404040   *                                *
0080   40404040 4C4C4040 40404040 40404040   40404040 40404040 40404040 40404040   *                                *
00A0   40404040 4C4C4040 4C404040 40404040   40404040 40404040 40404040 4C404040   *                                *
00C0   40404040 4C404040 40404040 40404040   40404040 40404040 40404040 4C404040   *                                *
00E0   40404040 4C404040 40404040 40404040   40404040 40404040 40404040 4C404040   *                                *
0100   F2873AF2 E3D0C6F1 F0080402 01F06FF1   03F276F3 57F418F5 5CF670F7 C7F87FF9   *2..STOH10....0.1.2.3.4.5.6&7.8.9*
0120   5FC13FC3 6CC57CC6 3CC83B01 63D36807   3EE46BE8 58700260 1C400000 11340101   *.A.C%E@F.H.J.L.P.U,Y$&.-. ......*
0140   A8C20101 C97408A3 90000200 F2810A7D   40000201 02C00101 481C0001 E6C1C201   *.B..........2..& .K...........B.*
0160   01C97D02 417C0341 7C879300 813F7C02   417C8653 35010011 78201389 2CCCCF01   *..&..@..@.....@..@..............*
0180   01C9F210 2FFC7C7C 7C008E73 F08E3003   8E7C908D 5C009000 880804F2 94CF9801   *..2..C@@...#0.&..@..*......2....*
01A0   0490F281 2EC20104 A0C08706 6EF0396F   00877C75 019F4C02 59018774 C25F6C00   *..2..B.......0....@...□.......%.*
01C0   67043502 CC116C06 6622B510 3D75019F   7A024AC0 87000484 00000000 CCCC0000   *......%.........................*
01E0   00000000 CCCC0000 C000C000 00000000   00000000 00000000 00000000 CCC000FF   *................................*
0200   2C01037F C3894000 F2103288 1000F290   052C0103 8106B8F0 04F21021 7C019F7C   *...... .2.....2.........0.2..@..@*
0220   69A77CF3 A5860104 F290043C EF03815E   009F9F5F 02A76C78 109FD090 5E680301   *..@3.....2..............%........A*
0240   00C20103 C59C0017 C86C0018 01750100   C08702F7 35010011 78801680 EF048840   *.B...*...%.........7....#...... *
0260   00F2941C 35C13318 C08702F7 88400DF2   900E0C01 027E0378 C0870004 8C1A8000   *.2.........7. .2.......#........*
0280   3C0102EF F2C10000 87000400 34020297   35020318 C20104A0 C087060E C2C102A3   *....2...............B.......B...*
02A0   74C87AF1 E6C07CC3 8C79E080 79F77FF2   103F5C01 7E6F7001 4CF2010E 35C20011   *...1W..T......7.2..*...&.□2......*
02C0   9CCC2F82 E20223C2 87030202 765C0134   04C08700 04C00000 026C0C82 CC7D014C   *....S..2..K..*..M.........%...&.□*
02E0   F2010434 CFC29000 4000A502 02C00101   477A4076 75107A34 010312C2 C1C2A374   *2....... ......... ........B....*
0300   C87A7401 73F287D4 34080316 31E601F7   F3E003C0 8702A3E0 03401676 C2CC0000   *.....2........W.73......... ...%..*
0320   00C00805 CCCC0000 00000000 00000000   00000000 00000000 00000000 CC000000   *................................*
0340   0C0C0C00 CCCC0000 C0000000 00000000   00000000 00000000 00000000 CCC00000   *................................*
0360   00000000 CCC3C4C9 D9E5C5E4 E2C7D3C1   D4E201FF C5071898 D6C81A80 4C404040   *.....CDIRVEUSGLAMS..EP..OH..    *
0380   4C404040 4C4C4040 40404040 40404040   40404040 40404040 40404040 4C404040   *                                *
03A0   40404040 4C4C4040 40404040 40404040   40404040 40404040 40404040 40404040   *                                *
03C0   40404040 4C4C4040 40404040 40404040   40404040 40404040 40404040 4C404040   *                                *
03E0   40404040 4C4C4040 40404040 40404040   40404040 40404040 40404040 4C404040   *                                *
0400   F1F2F3F4 F5F6F7F8 F9F07B7C 61E2E3E4   E5E6E7E8 E9506B6C D1D2D3D4 D5D6D7D8   *1234567890#@/STUVWXYZ+,%JKLMNOPQ*
0420   D9605B5C C1C2C3C4 C5C6C7C8 C94E4B7D   00000000 00000000 00000000 CCCC0000   *R-$*ABCDEFGHI..&................*
0440   0C000000 CCCC0000 00000000 00000000   00000000 00000000 00000000 CCCC0000   *................................*
0460   0C000000 CCCC0000 00000000 00000000   00000000 00000000 04A00000 C5C40084   *.............................M..*
0480   4200F12B CCCC0E00 A91400C3 00000009   E7000000 0E4C40A9 000E4003 740C0020   *..1........C....X....□ ... .....*
04A0   00280212 1F2204A0 084E1180 051AB040   A9001A80 00010000 200C2600 38167616   *.............. ...............*
04C0   28850000 7E0CA9F5 00510414 000F0016   761838F5 00027600 FFCBE2C5 C50800FF   *.......5...........5......SEE...*
04E0   600F7C00 CC5C03E2 C609E35C F0F5F0F6   F7F01B60 03F00208 00A96000 CC000000   *-.%..$CSORT*050670...0.........*
0500   0004A000 3402063A C2020518 8408008C   01E30484 8C008908 F1AE0100 E5800089   *........B.......T......1.......*
0520   F20108C2 C2C44035 10051888 8080F210   2788C089 F290038B 40B93401 C4CE3501   *2..B..........2.....2... .......*
0540   063A1C09 C4F209C2 0104A04C 0123063A   6C001489 6C012500 B88089F2 1C0A8841   *.....S.B...□....%...%......2....*
0560   39C09006 7A8C8189 2C01066D 8C893F89   F2900EAE 010087B8 808DF210 C46C0125   *................2.........2..%..*
0580   0C8A808D 6F40B9F2 9028AE00 80858805   B0F29007 F00057C0 87C000AC C6C4C8AC   *..... .2.........2..0........D..*
05A0   06CB02AC C6D209AC 02058C64 0107AC01   0900C201 04A09C01 E546893F E9F29052   *..K..KR..N...P..R.B.....V....2..*
05C0   AC01B000 4CC30802 80F2875C 00010003   04001AB0 00000000 00000000 CC000000   *....□....2.*....................*
05E0   00000000 CCCC0000 00000000 00000000   00001B14 19201638 180C26CC 1B801820   *................................*
```

Figure 3-12. Sample Program Storage Dump (Part 1 of 6)



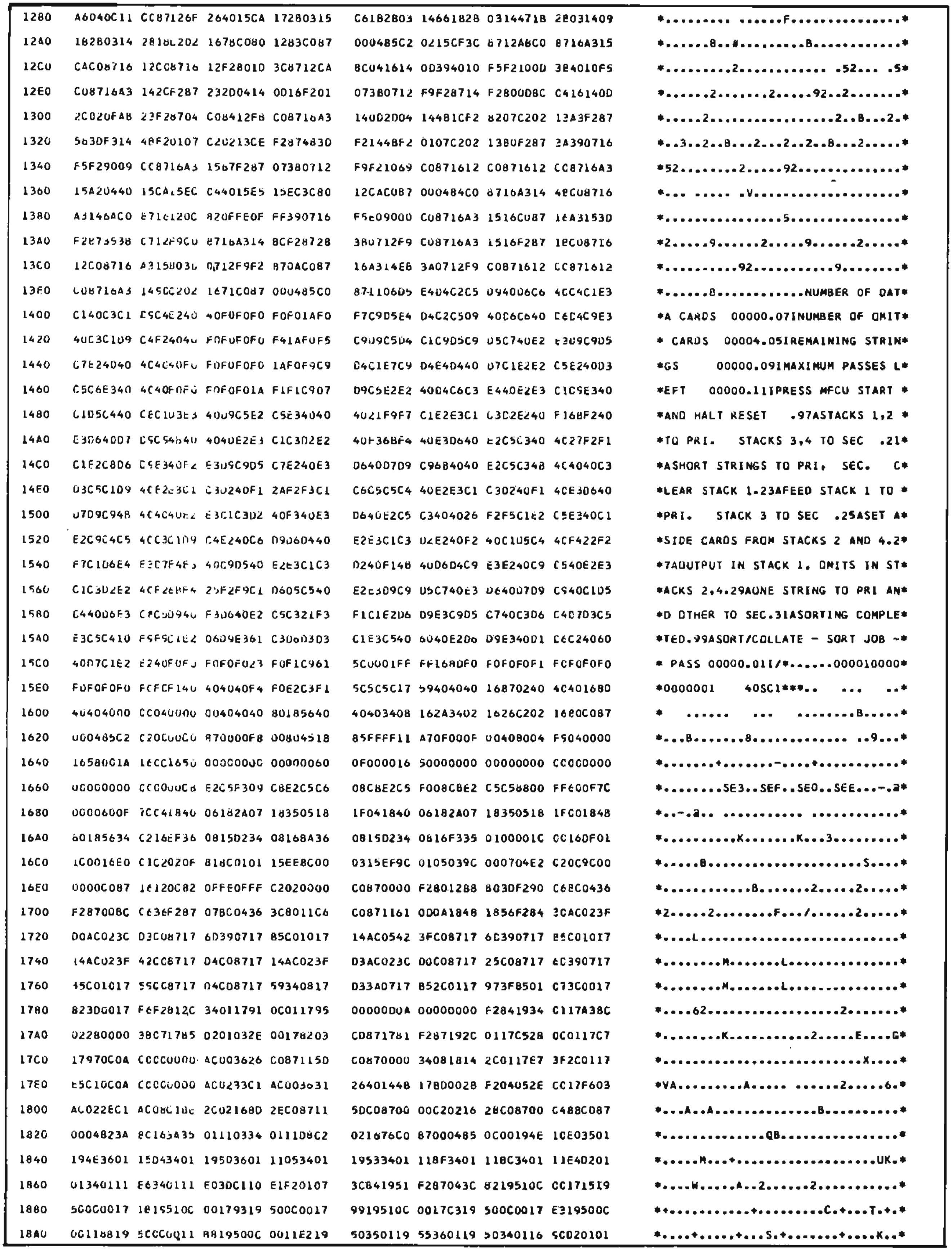
```
0600   18301840 185C17DC 1C982A00 1C241C84   16D48BC0 89AE0089 892C0006 2489E202   *... .+..........M...........S.*
0620   D96C0106 CA7C0207 4C000304 884C0109   05CE1D01 050306C2 0209E7F2 8126C087   *R%...2..□....□.......L.B..X2....*
0640   002CF287 1F0F0005 D5D5C04C 020705ED   C087002C 350105EF 0C01066D C5F10C14   *..2.....N..□................1..*
0660   05F105EA CCC2C5D4 C4C4C087 01003907   05D5C090 06453B80 05C5784D 14CC9007   *.1.....M.........N.......N. ....*
0680   A5790C14 F29C085C C135305E 01351B78   0414F290 0C5C011D 2C5C011B 3E5F011B   *....2..*.........2..*...*......*
06A0   30791C14 F29C2B5C C12B2D5C 01201D79   7F2BF210 194F002B 05CD4E00 2C05D07B   *....2..*...*.....2...........*
06C0   602DF290 C54EC120 002BC087 06AF5E01   202B5001 3516F202 03F00076 4CCC0304   *-.2.................2..0..□...*
06E0   68782014 F210045C 0003195C 0109355C   02072E4F 000705CD C087002C 5C012735   *....2..*...*...*..........*...*
0700   5F012730 F2B1045C 0129355C 0028317C   002A7D00 31F20103 7C012A5C C106017C   *....2..*...*...@..&..2..@..*...@*
0720   C2074C01 C9C5CE3C 0005D2C0 87002C35   0205CE1C 01075C29 7E02287C CC2A6C00   *..□.......K...........*....@..%.*
0740   2B008080 CE8FFF00 F2103CF2 84101E01   075C2BB0 8000F281 051E0119 5E27E202   *.......2..2.....*....2.....$.S.*
0760   01740229 78C42EC0 90073E4C 0129075C   7C00284E 00060029 786006F2 5C054E01   *..........□...*@........-.2....*
0780   C6002BC0 87C72B5E C1332775 02237810   14F29007 5C012533 F2870A7D 4C14F281   *................2..*...2..& .2.*
07A0   049C0909 35750121 35100402 74080F74   02117C06 02781003 F2900835 C209FD34   *..........K.....@.....2.......*
07C0   0109F0F2 E7063502 09EA3401 09EA6C01   01018401 017C000C 7B0700C0 87C00400   *...2..........%......@..#.......*
07E0   B40155C1 A2C8EC33 0209C465 01017880   02F29081 180007FA 0370A20B 79FF0A79   *...A......D......2.............*
0800   2408CC90 CA1F7901 0379400D F290139C   0112047B F0027806 037A0103 7A0304F2   *.......... .2......#0-#........2*
0820   87537A40 C2C1A408 32780303 F290117A   02024C01 060A0F78 8008F290 C37A0402   *... .A......2.....□........2.....*
0840   760304F2 5CC40C01 04127001 04790203   F296054C 02170A0F 8501019C C101014C   *...2..%...&.....2..□...........□*
0860   01010A0F EC010608 4C010105 F20106B4   0203F287 04C087D8 FC85020D CC8708FD   *....%.......2.....2.............*
0880   78A002F2 1CCC6502 00350101 78A002F2   905A2C03 0A100978 1C0DF290 C51C020A   *...2..........2.........2...:*
08A0   0F147B04 CCF21008 180008AE 0371A409   1C01086B 04F3A900 7B040D7A 1C023402   *.....2..............3..#.......*
08C0   09C4C202 C9C77010 03F21003 E2021078   0803F210 03E20208 790103F2 1CC97803   *.DB..G...2..S.....2..S.....2....*
08E0   04F21008 E2C2048E 0303CA14 C0870004   00B50155 760802F2 1099F287 CC340809   *.2..S................2..2.....*
0900   65650101 AC010105 F2817778 A002F210   71788002 F2104A9C 020907BC CC0E2C02   *........2.....2.....2...........*
0920   0A0F082C CC0A100E 3A030A0F 0F000A10   0A0EF202 0F0C000A 100A0E2F CCCA100E   *................2..............*
0940   3A010A0F 1E0C094A 0331A609 C61C0009   57033B03 0957F3A8 008C000E CACE7A80   *...........F........3........*
0960   02180009 67C370A2 03781008 F2101379   FF0A7924 0BF290A7 790303C0 1CC8227A   *...........2.......2.........*
0980   2C02C087 CF8C7408 0F740211 7A0802C0   87070B78 4002F210 07F10000 CC87098F   *.................. .2..1......*
09A0   78B80279 2CC07904 02F29C11 79010279   FF0CF210 08D20203 C0870004 CC750211   *#........2........2..K.........*
09C0   75100F09 E70A0000 000C0000 00000000   00000000 00000000 00001400 CCC21900   *....X.........................*
09E0   0C000600 CC0C0009 E709E709 E7001A80   00002009 FA1A0000 AA0009FA C9FA09FA   *........X.X.X...............*
0A00   00000000 CCC009F7 00000000 00001A80   FF000000 010A1700 0008FF00 2408922C   *.......X......................*
0A20   000A6A09 5CC10B0B 781D0AF2 10F37940   0A79040B F290E18C 03090A10 79CC0AF2   *...........2.3. ....2..........2*
0A40   900978C1 CAF21078 F2B7A678 800DF210   6F1C000A 5F033B02 0A5F3A01 CA5FF3A9   *.....2.#2.....2...............3.*
0A60   01C1A20A E1C1A00A 49600209 F2811679   0203F290 0870A40B 4F000A0A 1471A408   *.A../A......2.....2............*
0A80   BE00090A 1489280B 3D010A0D F2921A78   010DB97F 08F2105E F2810EBC C1067A01   *...........2........2..2.......*
0AA0   008C0110 CACFF287 624EC00C 0A14790F   0CF21014 89040A78 03047D09 CCF2927D   *......2..........2........&..2.&*
0AC0   F287B44E CCCCCA82 F2A07C8C 000E31A6   0A161800 0AD803F3 A0001C00 CAE80338   *2.......2.@..........Q.3.....Y..*
0AE0   070AE83A C1CAE8F3 A901C1A2 0AEAF287   4C8E0108 0A1C8980 08F29604 AEC00706   *..Y...Y3..A...2.□........2......*
0B00   8B03062C C3CA1009 7B01007A 04000C01   09850A1E C0870923 F06F3F8C CC0EF287   *........#...............0.....2.*
0B20   1C7A020D 180C082A 037C030B 4E000C0A   147D020C F2820675 0808F06F 6C78040D   *................&..2.....0.%#..*
0B40   70F702C0 E7C8757A 4102C087 0840F310   00F31800 34020C27 C2020870 B4C1B3B4   *#7........... 3..3......B.......*
0B60   0400F297 15F0F040 40404040 40404040   00000000 00010000 00000000 CCC0AC00   *..2..00           ...............*
0B80   B9008C01 8A850105 B0110398 4003F210   61880800 F21018B9 4703F290 E6A00006   *..............2./...2.....2.....*
0BA0   08F20262 ACCC3A06 6C000102 F28719B0   1903B92F 03F2904B B84007F2 5C4EA000   *-2......%...2........2... .2....*
0BC0   0608F202 41BA64BA AE0006B1 B9C007F2   1044AC00 68067118 01801903 E8C203F2   *..2............2...............2*
0BE0   10218880 C7F29031 F31884AC 01017AF2   87283842 048EF210 0D3A0104 E8EA208A   *.....2..3......2.......2........*
0C00   BC10B9BA CCC7BA06 07F28711 F31012AC   01019EF2 87078A12 BAAC0001 EA85040D   *.........2..3......2...........*
0C20   C2010001 C2C20001 F30101C0 87084E03   04040402 1E040410 02020250 C4C7070A   *B...B...3.....................*
```

Figure 3-12. Sample Program Storage Dump (Part 2 of 6)



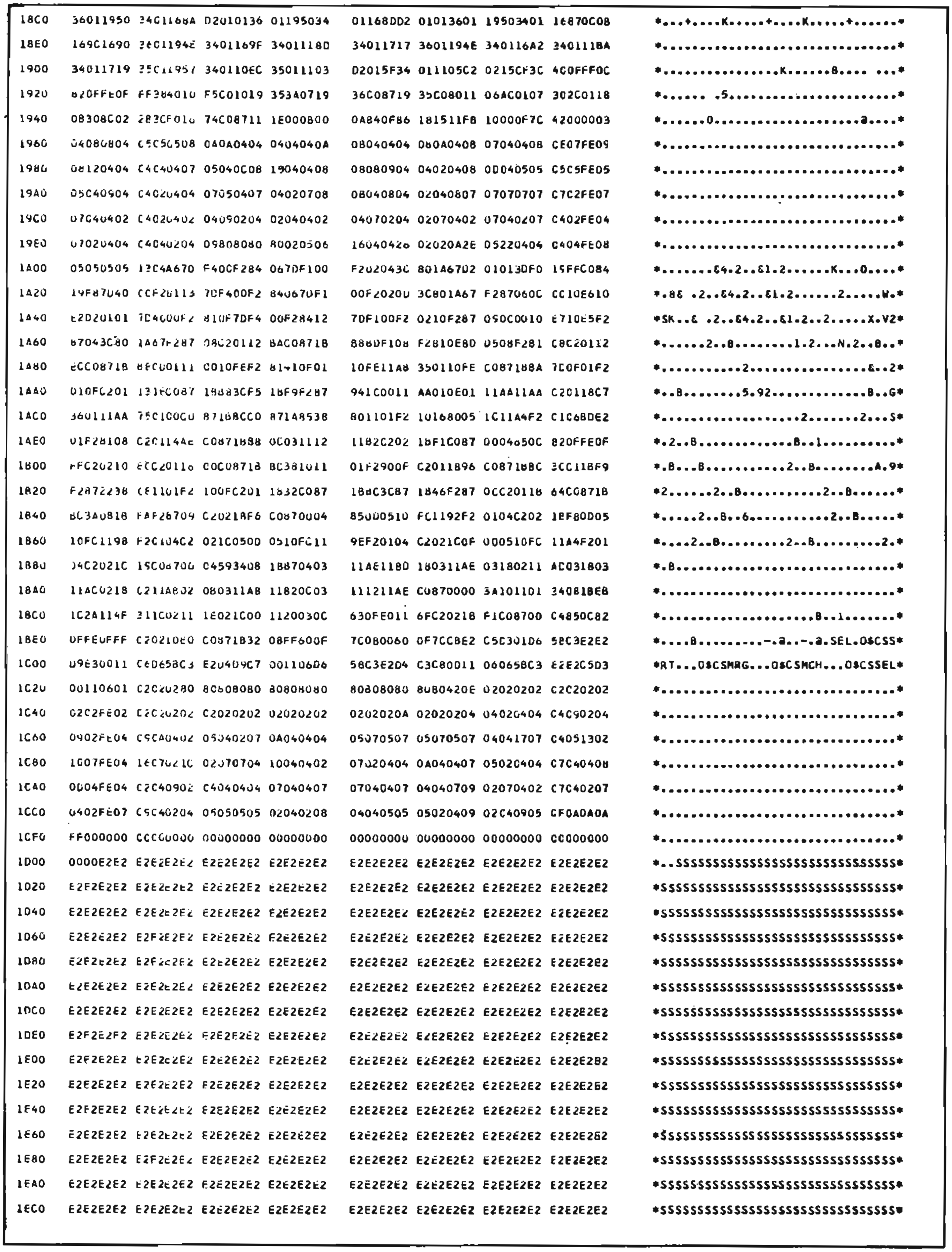
```
0C40   04040712 CA040402 040F1808 0704FE04   0E080404 04070404 04050704 C40A0404   *................................*
0C60   02040412 C2050802 C4070804 3404FE04   0705070A 0A040209 05020C04 C4CA2604   *................................*
0C80   02040604 C4400504 C5050407 0409FE02   08060204 02120404 02040420 CC150580   *................................*
0CA0   49041319 1A440408 13041304 0405FE05   15074A67 1005040C 07020504 C4C40204   *................................*
0CC0   02040208 C4040739 C5080413 1704FE28   04040404 18190508 08042513 C4070419   *................................*
0CE0   040D0E23 CE210C15 07160808 080EFE04   26152105 05040207 02040504 C50405FF   *................................*
0D00   B4080984 C1C06C40 CE840603 C0100E08   2C000D1A 113A070D 1AF1FF00 85C118C1   ... .......Q.........1.....A*
0D20   F00EA6AC CC1210B6 200FF290 142C010C   3F20BA02 1185010D 3A5F003F 1C5F0000   *0........2......................*
0D40   5F884C0F F2905280 F416887F 168E0016   0D9AB501 22600000 0CF28126 7EF000F2   *.. .2...4.................2...0.2*
0D60   90037501 C484C122 90001601 F2810F1C   000D7802 80F31488 0013C010 CC746C00   *............2........3........%.*
0D80   0C009C00 13C1C501 00BA7F16 2C010094   161C7F00 007F8A04 1188800F F2100C8D   *..........................2...*
0DA0   8C076910 CFF294E7 F28723BA 01113501   1C9C010D 01750104 71F50184 C11C8808   *.....2.X2...............5......*
0DC0   03FD8603 CC150E50 85011870 F502B804   11F2900A AE001213 8501227C CC008501   *.......+....5....2.........@....*
0DE0   1F850211 F2906F51 F62070F6 040C0106   122C010D F612F3F9 04C1F00E AEF88003   *....2...6..6.%......8.39.A0.....*
0E00   F2100989 CE03B801 11F29003 B806112C   000E1511 F1F9003C 040E2688 C811F210   *2........2...........19........2.*
0E20   043C080F 26760400 F290045C 070E0C0C   000E360E 267A0400 08800388 8CC3C010   *........2..*.....................*
0E40   00A58801 11F29C44 C1F00EA6 35080E0E   34080F98 B5010D70 6100F201 3E705C01   *.....2..A0.............&/.2..&*.*
0E60   F281107D 5CC1F201 2C580403 F2900EE2   0100C087 0004008A 204AD202 CC8A0203   *2..&+.2.....2..S..........K.....*
0E80   0C420E88 CFC3F290 C30A8003 88071185   01008510 09C08700 1A3E0000 CCC00000   *......2.........................*
0EA0   00000000 CC002C01 0FC81A34 020E00C2   020E9780 F5038408 40840101 85FF03F2   *.........H.....B....3... ......2*
0EC0   1C08C087 CC048016 0C027901 00C20216   28F29004 C0870DF0 8C410EC0 870E8C0F   *.............B...2..............*
0EE0   04050A0E CF16210B 12382008 16080A0E   020D0A04 04560504 2CFF0000 CCCC0000   *................................*
0F00   40F0F2F7 F7F4F2F3 F1F1F5D4 D6D9D9C9   E24005D1 40404040 40404040 40404040       423115MORRIS NJ              *
0F20   40404040 404040F2 F1F4F0F0 40404040   40404040 40404040 40404040 40404040         21400                      *
0F40   40404040 40404040 40404040 40404040   40404040 40404040 40404040 4040F3F3   *                             33*
0F60   E2E2E2E2 E2E2E2E2 E2E2E2E2 E2E2E2E2   E2E2E2E2 E2E2E2E2 E2E2E2E2 C0000000                       SSSSS....*
0F80   00000000 C0C00000 40404040 40404040   40404040 40404040 40404040 40404040   *........                        *
0FA0   40404040 40404040 40404040 40404040   40404040 40404040 40404040 40404040   *                                *
0FC0   40404040 40404040 40404040 40404040   40404040 40404040 40404040 40404040   *                                *
0FE0   F1404040 40404040 40404040 40404040   40404040 40404040 40404040 40404040                                    *
1000   E2E2E2E2 E2E2E2E2 E2E2E2E2 E2E2E2E2   E2E2E2E2 E2E2E2E2 E2E2E2E2 E2E2E2E2                   SSSSSSSSSSSSSSS*
1020   E2E2E2E2 E2E2E2E2 E2E2E2E2 E2E2E2E2   E2E2E2E2 E2E2E2E2 E2E2E2E2 E2E2E2E2                   SSSSSSSSSSSSSSS*
1040   E2E2E2E2 E2E2E2E2 E2E2E2E2 E2E2E2E2   E2E2E2E2 E2E2E2E2 E2E2E2E2 E2E2E2E2   *SSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSS*
1060   E2E2E2E2 E2E2E2E2 E2E2E2E2 E2E2E2E2   E2E2E2E2 E2E2E2E2 F0F5F0F6 F7F077E2   *SSSSSSSSSSSSSSSSSSSSSSSS050670.S*
1080   E2E2E2E2 E2E2E2E2 E2E2E2E2 E2E2E2E2   E2E2E2E2 E2E2E2E2 E2E2E2E2 E2E2E2E2    SSSSSSSSSSSSSSSSSSSSSSSSSSSSSSS*
10A0   E2E2E2E2 E2E2E2E2 E2E2E2E2 E2E2E2E2   E2E2F5F7 F0F260E4 E3F140C3 D6D7E8D9                  S5702-UT1 COPYR*
10C0   C9C7C8E3 40C9C2D4 40C3D6D9 D74B40F1   F9F7F040 40404040 40404040 40404040
10E0   0BC14040 40404040 40C08711 F8C0871F   280F0011 F0CF01E2 0609E340 4077FF77
1100   FF800F00 CF4FC202 15CF3C87 16F5AC01   072A2C01 160B2ABC 04283CD7 16742440                .....5.............P... *
1120   144B1024 4C146A1D 2440140D 1D3C0017   F6AC0BC1 C03C8712 1988803D 8C0536C0                .......6..A.............*
1140   871150BA 8C3D8C07 36C08711 5D3C8012   19308712 21C00117 14C08717 59340811   *................................*
1160   C0A8663D F2102DC2 0216286C F0008CF1   116C0010 1605C087 0C003502 16383501   *....2..B....0..1................*
1180   11036C5F 5F5FC087 11C10C0A 18480F86   C08711BD C2021628 8CF8008C CC101605   *..%.......A..........B....B......*
11A0   8CF911C0 87C00035 02163835 0111036C   5F5F5FC0 8711C10C 0A18560F 8EC08711   *.9..............%.......A........*
11C0   40340812 17F2870D C2021667 C0870004   853C8711 C6C2010F 00C20215 CF3C000F   *.....2..B............FB...B......*
11E0   870C0A0F 8ECF6790 C1010100 81121838   02162EC0 101218C0 8710EDC2 C215CF38   *...........................B....*
1200   8010F5F2 5C052640 142C17C0 8716F426   40140D17 C08711B7 F287052C C111C025   *..52... ......4. ........2.......*
1220   F287193C 87122188 803DF210 07AC023F   D3F28704 AC023FD0 C08711BD 3C801221   *2..........2......L2.............*
1240   3C871219 A440C017 2D04140D 0CF20215   2004142C 0CF2020D C202166C CC870004   *..... ........2........2..B..%....*
1260   85C08711 C63C400F FF0C820F FE0FFF2D   04144B0C F2041126 40146A17 AC04110C   *...... ..............2... ........*
```

Figure 3-12. Sample Program Storage Dump (Part 3 of 6)



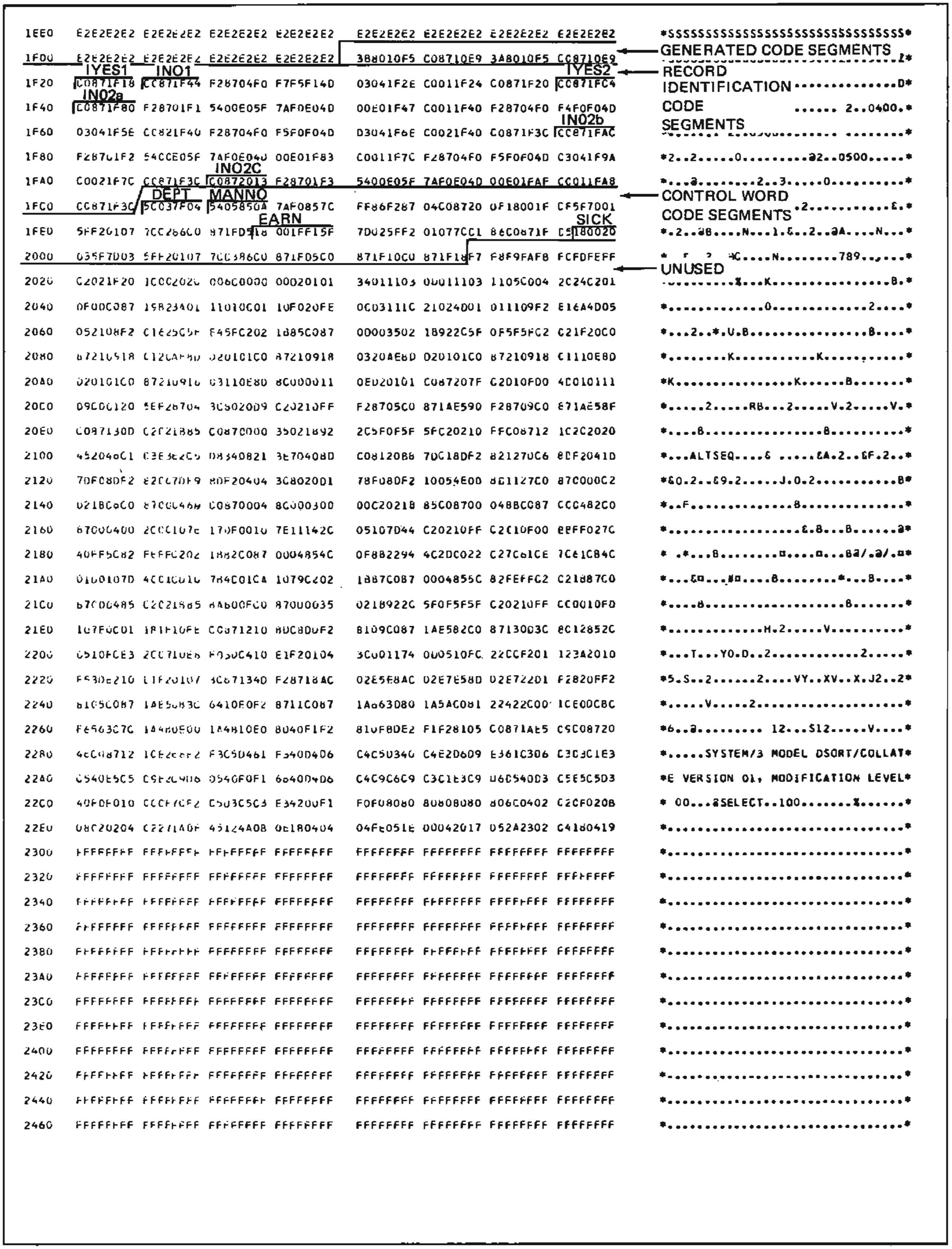
```
1280   A6040C11 CC87126F 264015CA 17280315   C6182803 14661828 0314471B 28031409   *......... ......F...............*
12A0   18280314 2818L202 167BC080 1283C087   000485C2 0215CF3C 8712A6C0 8716A315   *......8..#.........B............*
12C0   CAC08716 12C08716 12F2801D 3C8712CA   8C041614 00394010 F5F2100D 3B4010F5   *.........2............ .52... .5*
12E0   C08716A3 142CF287 232D0414 0016F201   073B0712 F9F28714 F2800D8C C416140D   *......2.......2.....92..2.......*
1300   2C020FAB 23F28704 C08412F8 C08716A3   140D2D04 14481CF2 8207C202 13A3F287   *.....2.................2..B...2.*
1320   563DF314 4BF20107 C20213CE F287483D   F2144BF2 0107C202 138DF287 2A390716   *..3..2..B...2...2..2..B...2.....*
1340   F5F29009 CC8716A3 1567F287 07380712   F9F21069 C0871612 C0871612 CC8716A3   *52........2.....92..............*
1360   15A20440 15CA15EC C44015E5 15EC3C80   12CAC087 000484C0 8716A314 4BC08716   *... ..... .V....................*
1380   A3146AC0 8716120C 820FFE0F FF390716   F5E09000 C08716A3 1516C087 16A3153D   *................5...............*
13A0   F2873538 C712F9C0 8716A314 8CF28728   3B0712F9 C08716A3 1516F287 1EC08716   *2.....9......2.....9......2.....*
13C0   12C08716 A315803D 0712F9F2 870AC087   16A314E6 3A0712F9 C0871612 CC871612   *..........92...........9........*
13E0   C08716A3 145CC202 1671C087 000485C0   871106D5 E4D4C2C5 D94006C6 40C4C1E3   *......8............NUMBER OF DAT*
1400   C140C3C1 D9C4E240 40F0F0F0 F0F01AF0   F7C9D5E4 D4C2C509 40D6C640 D6D4C9E3   *A CARDS  00000.07INUMBER OF OMIT*
1420   40C3C1D9 C4E24040 F0F0F0F0 F41AF0F5   C9D9C5D4 C1C9D5C9 D5C740E2 E3D9C9D5   * CARDS  00004.05IREMAINING STRIN*
1440   C7E24040 404040F0 F0F0F0F0 1AF0F9C9   D4C1E7C9 D4E4D440 D7C1E2E2 C5E240D3   *GS     00000.09IMAXIMUM PASSES L*
1460   C5C6E340 4040F0F0 F0F0F01A F1F1C907   D9C5E2E2 40D4C6C3 E440E2E3 C1D9E340   *EFT   00000.11IPRESS MFCU START *
1480   C1D5C440 C8C1D3E3 40D9C5E2 C5E34040   4021F9F7 C1E2E3C1 C3D2E240 F16BF240   *AND HALT RESET   .97ASTACKS 1,2 *
14A0   E3D64007 D9C94B40 4040E2E3 C1C3D2E2   40F36BF4 40E3D640 E2C5C340 4027F2F1   *TO PRI.   STACKS 3,4 TO SEC  .21*
14C0   C1E2C8D6 D9E340F2 E3D9C9D5 C7E240E3   D64007D9 C96B4040 E2C5C34B 404040C3   *ASHORT STRINGS TO PRI,  SEC.   C*
14E0   D3C5C1D9 40E2E3C1 C3D240F1 2AF2F3C1   C6C5C5C4 40E2E3C1 C3D240F1 40E3D640   *LEAR STACK 1.23AFEED STACK 1 TO *
1500   D7D9C94B 404040E2 E3C1C3D2 40F340E3   D640E2C5 C3404026 F2F5C1E2 C5E340C1   *PRI.   STACK 3 TO SEC  .25ASET A*
1520   E2C9C4C5 40C3C1D9 C4E240C6 D9D6D440   E2E3C1C3 D2E240F2 40C1D5C4 40F422F2   *SIDE CARDS FROM STACKS 2 AND 4.2*
1540   F7C1D6E4 E3D7E4E3 40C9D540 E2E3C1C3   D240F14B 40D6D4C9 E3E240C9 D540E2E3   *7AOUTPUT IN STACK 1. OMITS IN ST*
1560   C1C3D2E2 40F26BF4 25F2F9C1 D6D5C540   E2E3D9C9 D5C740E3 D640D7D9 C940C1D5   *ACKS 2,4.29AONE STRING TO PRI AN*
1580   C440D6E3 C8C5D940 E3D640E2 C5C321F3   F1C1E2D6 D9E3C9D5 C740C3D6 D4D7D3C5   *D OTHER TO SEC.31ASORTING COMPLE*
15A0   E3C5C410 F9F9C1E2 D6D9E361 C3D6D3D3   C1E3C540 6040E2D6 D9E340D1 D6C24060   *TED.99ASORT/COLLATE - SORT JOB -*
15C0   40D7C1E2 E240F0F0 F0F0F023 F0F1C961   5C0001FF FF168DF0 F0F0F0F1 F0F0F0F0   * PASS 00000.01I/*......000010000*
15E0   F0F0F0F0 F0F0F140 404040F4 F0E2C3F1   5C5C5C17 59404040 16870240 4C401680   *0000001    40SC1***..   ...  ..*
1600   40404000 CC040000 00404040 80185640   40403408 162A3402 1626C202 1680C087   *   ......   ...   ........B.....*
1620   000485C2 C200C0C0 870000F8 00804518   85FFFF11 A70F000F 00408004 F5040000   *...B.......8.............. ..9...*
1640   1658001A 16CC1650 0000000C 00000060   0F000016 50000000 00000000 CC000000   *.......+.......-....+...........*
1660   00000000 CC0000C8 E2C5F309 C8E2C5C6   08C8E2C5 F008C8E2 C5C58800 FF600F7C   *.........SE3..SEF..SEO..SEE...¬.@*
1680   0000600F 7CC41840 06182A07 18350518   1F041840 06182A07 18350518 1FC0184B   *..¬.@.. ........... ............*
16A0   60185634 C216EF36 08150234 08168A36   08150234 0816F335 0100001C 0C160F01   *...........K........K...3..........*
16C0   1C0016E0 C1C2020F 818C0101 15EE8C00   0315EF9C 0105039C 000704E2 C20C9C00   *.....B.....................S.....*
16E0   0000C087 16120C82 0FFE0FFF C2020000   C0870000 F2801288 803DF290 C6BC0436   *..............B.......2.....2.....*
1700   F287008C C636F287 07BC0436 3C8011C6   C0871161 0D0A1848 1856F284 3CAC023F   *2.....2.........F.../......2.....*
1720   D0AC023C D3C08717 6D390717 85C01017   14AC0542 3FC08717 6C390717 85C01017   *....L.............................*
1740   14AC023F 42C08717 D4C08717 14AC023F   D3AC023C D0C08717 25C08717 6C390717   *........M.......L................*
1760   45C01017 55C08717 04C08717 59340817   D33A0717 852C0117 973F8501 C73C0017   *........M.......L................*
1780   823D0017 F6F2812C 34011791 0C011795   00000D0A 00000000 F2841934 C117A38C   *....62.....................2.......*
17A0   02280000 38C71785 0201032E 00178203   CD871781 F287192C 0117C528 0C0117C7   *.........K............2.....E....G*
17C0   17970C0A CCCC0000 AC003626 C087115D   C0870000 34081814 2C0117E7 3F2C0117   *..........................X.....*
17E0   E5C10C0A CCCC0000 AC0233C1 AC003631   26401448 17BD0028 F204052E CC17F603   *VA.........A..... ......2.....6.*
1800   AC022EC1 AC08C18C 2C02168D 2EC08711   5DC08700 00C20216 28C08700 C488C087   *...A..A...............B..........*
1820   0004823A 8C163A35 01110334 0111D8C2   021676C0 87000485 0C00194E 10E03501   *...............QB...............*
1840   194E3601 15D43401 19503601 11053401   19533401 118F3401 118C3401 11E4D201   *.....M...+....................UK.*
1860   01340111 E6340111 E03DC110 E1F20107   3C841951 F287043C 8219510C CC171519   *....W......A..2......2...........*
1880   50C00017 1E19510C 00179319 500C0017   9919510C 0017C319 500C0017 E319500C   *+..........+..........C.+...T.+.*
18A0   0C118819 5CCC0011 8819500C 0011E219   50350119 55360119 50340116 5C020101   *....+......+...S.+........+....K..*
```

Figure 3–12. Sample Program Storage Dump (Part 4 of 6)



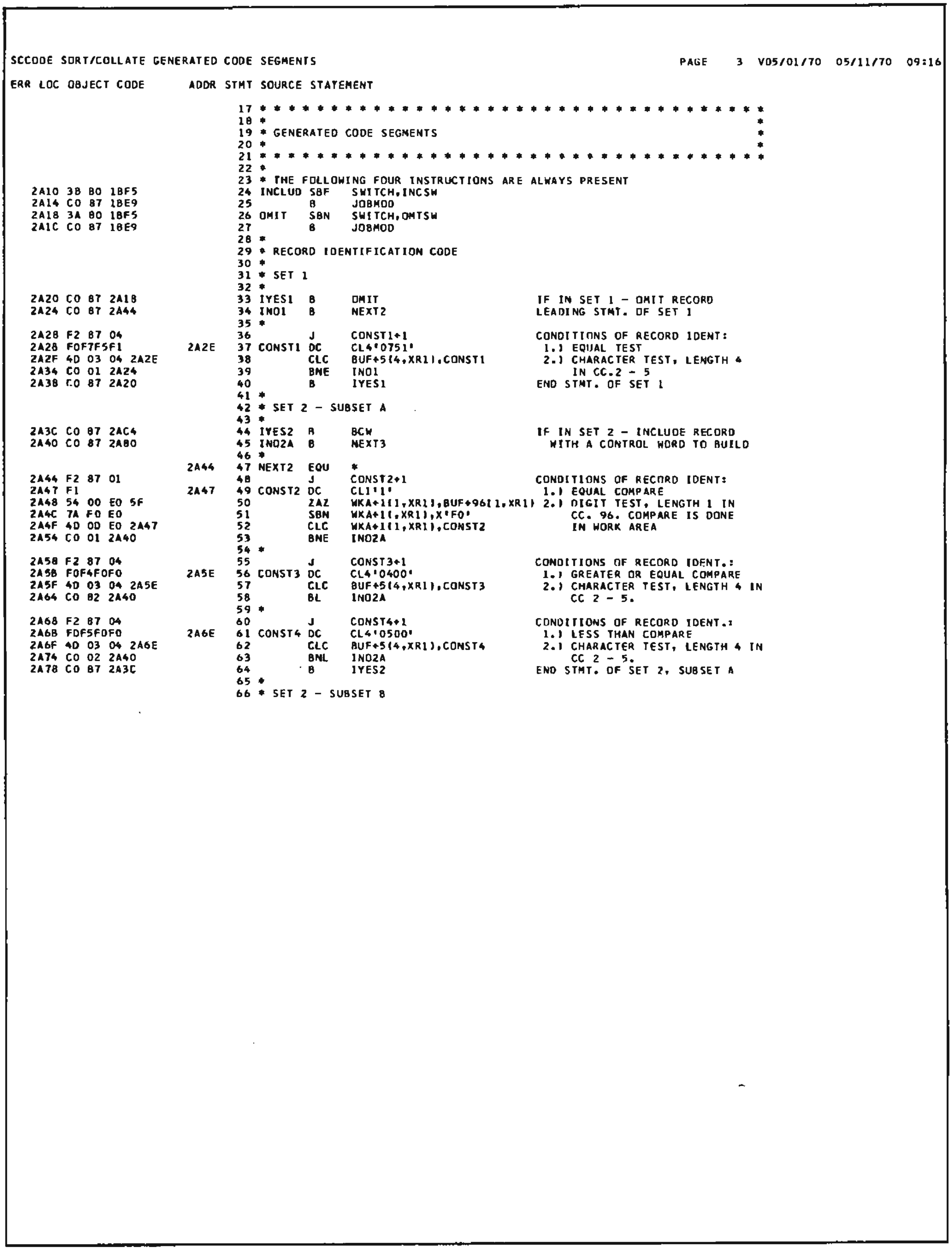
```
18C0   36011950 34C1168A D2010136 01195034   01168DD2 01013601 19503401 16870C08    *...+....K.....+....K.....+......*
18E0   169C1690 36C1194E 3401169F 34011180   34011717 3601194E 340116A2 340111BA    *................................*
1900   34011719 35C11957 340110EC 35011103   D2015F34 011105C2 0215CF3C 4C0FFF0C    *................K......B.... ...*
1920   620FFE0F FF384010 F5C01019 35340719   36C08719 35C08011 06AC0107 302C0118    *...... .5.......................*
1940   08308C02 2B3CF016 74C08711 1E000800   0A840F86 181511F8 10000F7C 42000003    *......0....................@....*
1960   04080804 C5C5C508 040A0404 0404040A   08040404 080A0408 07040408 CE07FE09    *................................*
1980   08120404 C4C40407 05040C08 19040408   08080904 04020408 0D040505 C5C5FE05    *................................*
19A0   05C40904 C4020404 07050407 04020708   08040804 02040807 07070707 C7C2FE07    *................................*
19C0   07C40402 C4020402 04090204 02040402   04070204 02070402 07040207 C402FE04    *................................*
19E0   07020404 C4040204 09808080 80020506   16040426 02020A2E 05220404 C404FE08    *................................*
1A00   05050505 12C4A670 F40CF284 067DF100   F202043C 801A67D2 01013DF0 15FFC084    *.......&4.2..&1.2......K...0....*
1A20   19F87040 CCF26113 7DF400F2 84067DF1   00F20200 3C801A67 F287060C CC10E610    *.8& .2..&4.2..&1.2......2.....W.*
1A40   E2D20101 7D4000F2 810F7DF4 00F28412   7DF100F2 0210F287 050C0010 E710E5F2    *SK..& .2..&4.2..&1.2..2.....X.V2*
1A60   87043C80 1A67F287 08C20112 BAC0871B   888DF108 F2810E8D D508F281 C8C20112    *......2..B........1.2...N.2..B..*
1A80   ECC0871B 8FC00111 C010FEF2 81410F01   10FE11A8 350110FE C08718BA 7C0F01F2    *...........2................&..2*
1AA0   010FC201 131FC087 18883CF5 18F9F287   941C0011 AA010E01 11AA11AA C20118C7    *..B........5.92.............B..G*
1AC0   360111AA 75C100C0 8718BCC0 87148538   801101F2 10168005 1C11A4F2 C1C6BDE2    *...................2.......2...S*
1AE0   01F28108 C2C114AE C0871B88 0C031112   11B2C202 1BF1C087 0004650C 820FFE0F    *.2..B.............B..1..........*
1B00   FFC20210 ECC20116 C0C0871B BC381011   01F2900F C2011896 C0871BBC 3CC11BF9    *.B...B...........2..B........A.9*
1B20   F2872238 CE1101F2 100FC201 1832C087   18BC3C87 1846F287 0CC2011B 64C0871B    *2......2..B...........2..B......*
1B40   BC3A0818 FAF26709 C2021BF6 C0870004   85000510 FC1192F2 0104C202 1BF80D05    *.....2..B..6...........2..B.....*
1B60   10FC1198 F2C104C2 021C0500 0510FC11   9EF20104 C2021C0F 000510FC 11A4F201    *....2..B.........2..B.........2.*
1B80   34C2021C 15C08700 04593408 1B870403   11AE1180 180311AE 03180211 AC031803    *.B..............................*
1BA0   11AC0218 C211AB02 080311AB 11820C03   111211AE C0870000 3A101101 34081BEB    *................................*
1BC0   1C2A114F 311C0211 1E021C00 1120030C   630FE011 6FC20218 F1C08700 C4850C82    *.....................B..1.......*
1BE0   0FFE0FFF C20210E0 C0871B32 08FF600F   7C080060 0F7CCBE2 C5D301D6 5BC3E2E2    *....B.........-.@..-.@.SEL.O$CSS*
1C00   D9E30011 C6D65BC3 E2D4D9C7 001106D6   5BC3E2D4 C3C80011 06065BC3 E2E2C5D3    *RT...O$CSMRG...O$CSMCH...O$CSSEL*
1C20   00110601 C2C20280 80808080 80808080   80808080 80B0420E 02020202 C2C20202    *................................*
1C40   C2C2FE02 C2C20202 C2020202 02020202   0202020A 02020204 04020404 C4C90204    *................................*
1C60   0902FE04 C9C40402 05040207 0A040404   05070507 05070507 04041707 C4051302    *................................*
1C80   1C07FE04 1CC70210 02070704 10040402   07020404 0A040407 05020404 C7C40408    *................................*
1CA0   0C04FE04 C2C40902 C4040404 07040407   07040407 04040709 02070402 C7C40207    *................................*
1CC0   0402FE07 C9C40204 05050505 02040208   04040505 05020409 02C40905 CF0A0A0A    *................................*
1CE0   FF000000 CCC00000 00000000 00000000   00000000 00000000 00000000 C0000000    *................................*
1D00   0000E2E2 E2E2E2E2 E2E2E2E2 E2E2E2E2   E2E2E2E2 E2E2E2E2 E2E2E2E2 E2E2E2E2    *..SSSSSSSSSSSSSSSSSSSSSSSSSSSSSS*
1D20   E2E2E2E2 E2E2E2E2 E2E2E2E2 E2E2E2E2   E2E2E2E2 E2E2E2E2 E2E2E2E2 E2E2E2E2    *SSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSS*
1D40   E2E2E2E2 E2E2E2E2 E2E2E2E2 E2E2E2E2   E2E2E2E2 E2E2E2E2 E2E2E2E2 E2E2E2E2    *SSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSS*
1D60   E2E2E2E2 E2E2E2E2 E2E2E2E2 E2E2E2E2   E2E2E2E2 E2E2E2E2 E2E2E2E2 E2E2E2E2    *SSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSS*
1D80   E2E2E2E2 E2E2E2E2 E2E2E2E2 E2E2E2E2   E2E2E2E2 E2E2E2E2 E2E2E2E2 E2E2E2E2    *SSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSS*
1DA0   E2E2E2E2 E2E2E2E2 E2E2E2E2 E2E2E2E2   E2E2E2E2 E2E2E2E2 E2E2E2E2 E2E2E2E2    *SSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSS*
1DC0   E2E2E2E2 E2E2E2E2 E2E2E2E2 E2E2E2E2   E2E2E2E2 E2E2E2E2 E2E2E2E2 E2E2E2E2    *SSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSS*
1DE0   E2E2E2E2 E2E2E2E2 E2E2E2E2 E2E2E2E2   E2E2E2E2 E2E2E2E2 E2E2E2E2 E2E2E2E2    *SSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSS*
1E00   E2E2E2E2 E2E2E2E2 E2E2E2E2 E2E2E2E2   E2E2E2E2 E2E2E2E2 E2E2E2E2 E2E2E2E2    *SSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSS*
1E20   E2E2E2E2 E2E2E2E2 E2E2E2E2 E2E2E2E2   E2E2E2E2 E2E2E2E2 E2E2E2E2 E2E2E2E2    *SSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSS*
1E40   E2E2E2E2 E2E2E2E2 E2E2E2E2 E2E2E2E2   E2E2E2E2 E2E2E2E2 E2E2E2E2 E2E2E2E2    *SSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSS*
1E60   E2E2E2E2 E2E2E2E2 E2E2E2E2 E2E2E2E2   E2E2E2E2 E2E2E2E2 E2E2E2E2 E2E2E2E2    *SSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSS*
1E80   E2E2E2E2 E2E2E2E2 E2E2E2E2 E2E2E2E2   E2E2E2E2 E2E2E2E2 E2E2E2E2 E2E2E2E2    *SSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSS*
1EA0   E2E2E2E2 E2E2E2E2 E2E2E2E2 E2E2E2E2   E2E2E2E2 E2E2E2E2 E2E2E2E2 E2E2E2E2    *SSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSS*
1EC0   E2E2E2E2 E2E2E2E2 E2E2E2E2 E2E2E2E2   E2E2E2E2 E2E2E2E2 E2E2E2E2 E2E2E2E2    *SSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSS*
```

Figure 3-12. Sample Program Storage Dump (Part 5 of 6)


GENERATED CODE SEGMENTS
RECORD IDENTIFICATION CODE SEGMENTS
CONTROL WORD CODE SEGMENTS
UNUSED
IYES1
INO1
IYES2
INO2a
INO2b
INO2C
DEPT
MANNO
EARN
SICK

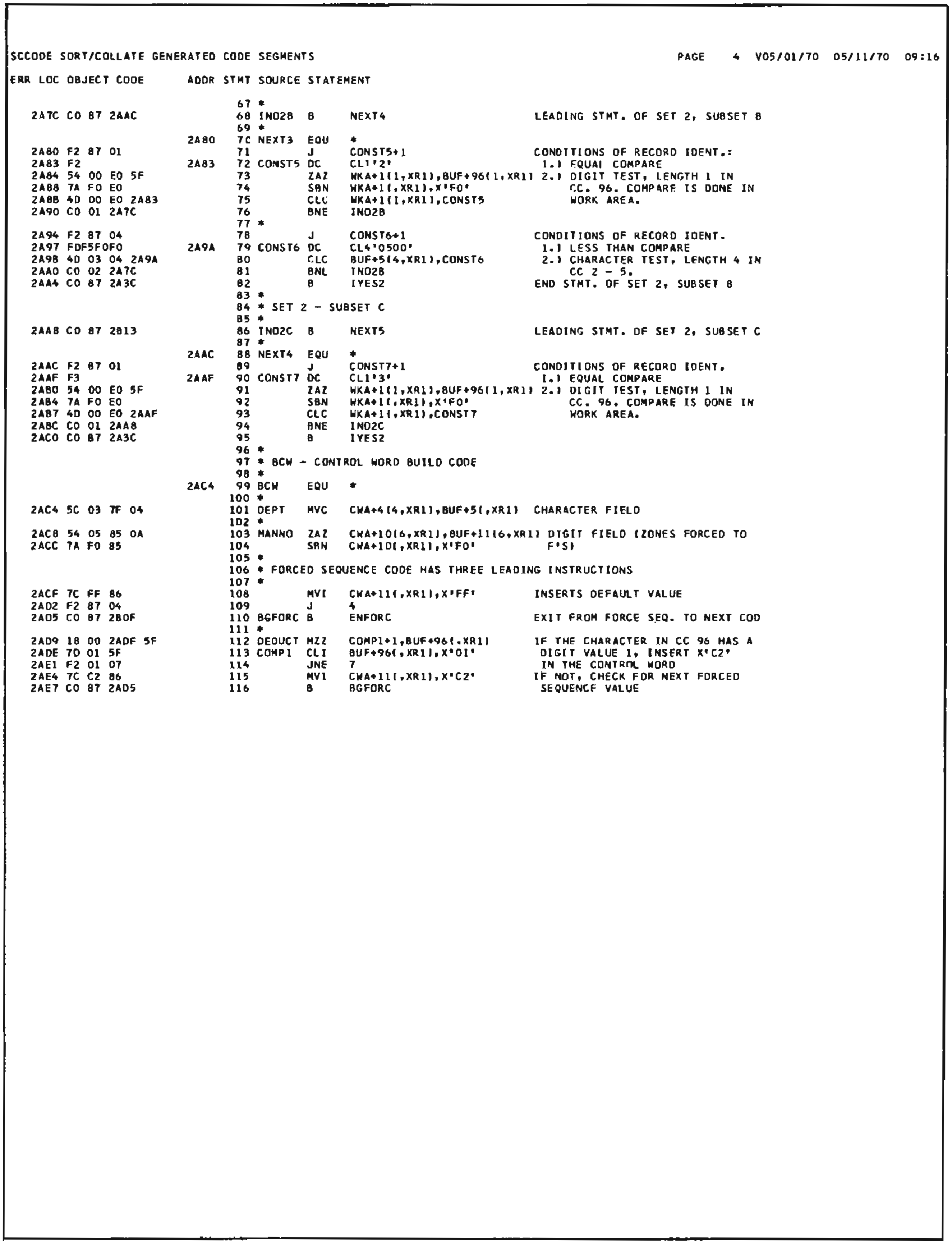
```
1EE0   E2E2E2E2 E2E2E2E2 E2E2E2E2 E2E2E2E2   E2E2E2E2 E2E2E2E2 E2E2E2E2 E2E2E2E2     *SSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSS*
1F00   E2E2E2E2 E2E2E2E2 E2E2E2E2 E2E2E2E2   3B8010F5 C08710E9 3A8010F5 CC8710E9     *...............................2*
1F20   CC871F18 CC871F44 F28704F0 F7F5F14D   03041F2E C0011F24 C0871F20 CC871FC4     *...............................D*
1F40   CC871F80 F28701F1 5400E05F 7AF0E04D   00E01F47 C0011F40 F28704F0 F4F0F04D     *...................... 2..0400.*
1F60   03041F5E CC821F40 F28704F0 F5F0F04D   D3041F6E C0021F40 C0871F3C CC871FAC     *................................*
1F80   F28701F2 54CCE05F 7AF0E040 00E01F83   C0011F7C F28704F0 F5F0F04D C3041F9A     *2..2.....0.........@2..0500.....*
1FA0   C0021F7C CC871F3C C0872013 F28701F3   5400E05F 7AF0E04D 00E01FAF CC011FA8     *...@........2..3.....0..........*
1FC0   CC871F3C 5C037F04 5405850A 7AF0857C   FF86F287 04C08720 0F18001F CF5F7001     *.....................2.........£.*
1FE0   5FF20107 7CC266C0 871FD518 001FF15F   7D025FF2 01077CC1 86C0871F C5180020     *.2..@B....N....1.&..2..@A....N...*
2000   035F7D03 5FF20107 7CC386C0 871FD5C0   871F10C0 871F18F7 F8F9FAFB FCFDFEFF     *.'..2..@C....N.........789......*
2020   C2021F20 1C002020 006C0000 00020101   34011103 00011103 1105C004 2C24C201     *.........%...K................B.*
2040   0F00C087 19B23A01 11010C01 10F020FE   0C03111C 21024D01 011109F2 E16A4D05     *.............0..............2....*
2060   052108F2 C1625C5F F45FC202 1885C087   00003502 18922C5F 0F5F5FC2 C21F20C0     *...2..*.U.B................B....*
2080   87210518 C120AF8D 020101C0 87210918   0320AE8D 020101C0 87210918 C1110E8D     *........K...........K...........*
20A0   020101C0 87210918 03110E80 8C000011   0E020101 C087207F C2010F00 4C010111     *K...............K......B........*
20C0   09C00120 5EF28704 3C5020D9 C20210FF   F28705C0 871AE590 F28709C0 871AE58F     *.....2.....RB...2.....V.2.....V.*
20E0   C087130D C2021885 C0870000 35021892   2C5F0F5F 5FC20210 FFC08712 1C2C2020     *....B...............B...........*
2100   452040C1 C3E3E2C5 D8340821 3E70408D   C0812086 70C18DF2 82127006 8DF2041D     *...ALTSEQ....& .....&A.2..&F.2..*
2120   7DF08DF2 820C7DF9 8DF20404 3C802001   78F08DF2 10054E00 8C1127C0 87C000C2     *&0.2..&9.2.....J.0.2...........B*
2140   021BC6C0 87C0C46B C0870004 8C000300   00C2021B 85C08700 048BC087 CCC482C0     *..F...............B.............*
2160   87000400 2CCC107E 170F0010 7E11142C   05107D44 C20210FF C2C10F00 8EFF027C     *..................&.B...B......@*
2180   40FF5C82 FEFFC202 1882C087 0004854C   0F882294 4C2DC022 C27C61CE 7C61C84C     * .*...B.........¤....¤...B@/.@/.¤*
21A0   01D0107D 4CC1C010 784C01CA 1079C202   1887C087 0004855C 82FEFFC2 C21887C0     *...&¤...#¤....B.........*...B....*
21C0   87C00485 C2021885 8A600FC0 87000035   0218922C 5F0F5F5F C20210FF CC0010FD     *....B....................B.......*
21E0   1C7F0C01 1F1F10FE C0871210 8DC8D0F2   8109C087 1AE582C0 8713003C 8C12852C     *..............H.2.....V.........*
2200   0510FCE3 2CC710E8 F050C410 E1F20104   3C001174 000510FC 22CCF201 123A2010     *...T...YO.D..2..............2.....*
2220   F530E210 C1F20107 3C871340 F28718AC   02E5E8AC 02E7E58D 02E722D1 F2820FF2     *5.S..2.......2....VY..XV..X.J2..2*
2240   8105C087 1AE5083C 6410F0F2 8711C087   1A863080 1A5AC081 22422C00 1CE0DCBC     *.....V.....2....................*
2260   F6563C7C 1A480E00 1A4810E0 8040F1F2   810F8DE2 F1F28105 C0871AE5 C9C08720     *6..@.......... 12...S12.....V....*
2280   4CC08712 1CE2E8F2 F3C5D461 F540D406   C4C50340 C4E2D609 E361C306 C3D3C1E3     *.....SYSTEM/3 MODEL DSORT/COLLAT*
22A0   C540E5C5 D9E2C9D6 D540F0F1 6B40D4D6   C4C9C6C9 C3C1E3C9 D6D540D3 C5E5C5D3     *E VERSION 01, MODIFICATION LEVEL*
22C0   40F0F010 CCCF7CF2 C5D3C5C3 E34200F1   F0F08080 80808080 806C0402 C2CF020B     * 00...@SELECT..100.......%......*
22E0   08C20204 C2271A0F 45124A08 0E180404   04FE051E 00042017 052A2302 C4180419     *................................*
2300   FFFFFFFF FFFFFFFF FFFFFFFF FFFFFFFF   FFFFFFFF FFFFFFFF FFFFFFFF FFFFFFFF     *................................*
2320   FFFFFFFF FFFFFFFF FFFFFFFF FFFFFFFF   FFFFFFFF FFFFFFFF FFFFFFFF FFFFFFFF     *................................*
2340   FFFFFFFF FFFFFFFF FFFFFFFF FFFFFFFF   FFFFFFFF FFFFFFFF FFFFFFFF FFFFFFFF     *................................*
2360   FFFFFFFF FFFFFFFF FFFFFFFF FFFFFFFF   FFFFFFFF FFFFFFFF FFFFFFFF FFFFFFFF     *................................*
2380   FFFFFFFF FFFFFFFF FFFFFFFF FFFFFFFF   FFFFFFFF FFFFFFFF FFFFFFFF FFFFFFFF     *................................*
23A0   FFFFFFFF FFFFFFFF FFFFFFFF FFFFFFFF   FFFFFFFF FFFFFFFF FFFFFFFF FFFFFFFF     *................................*
23C0   FFFFFFFF FFFFFFFF FFFFFFFF FFFFFFFF   FFFFFFFF FFFFFFFF FFFFFFFF FFFFFFFF     *................................*
23E0   FFFFFFFF FFFFFFFF FFFFFFFF FFFFFFFF   FFFFFFFF FFFFFFFF FFFFFFFF FFFFFFFF     *................................*
2400   FFFFFFFF FFFFFFFF FFFFFFFF FFFFFFFF   FFFFFFFF FFFFFFFF FFFFFFFF FFFFFFFF     *................................*
2420   FFFFFFFF FFFFFFFF FFFFFFFF FFFFFFFF   FFFFFFFF FFFFFFFF FFFFFFFF FFFFFFFF     *................................*
2440   FFFFFFFF FFFFFFFF FFFFFFFF FFFFFFFF   FFFFFFFF FFFFFFFF FFFFFFFF FFFFFFFF     *................................*
2460   FFFFFFFF FFFFFFFF FFFFFFFF FFFFFFFF   FFFFFFFF FFFFFFFF FFFFFFFF FFFFFFFF     *................................*
```

Figure 3-12. Sample Program Storage Dump (Part 6 of 6)



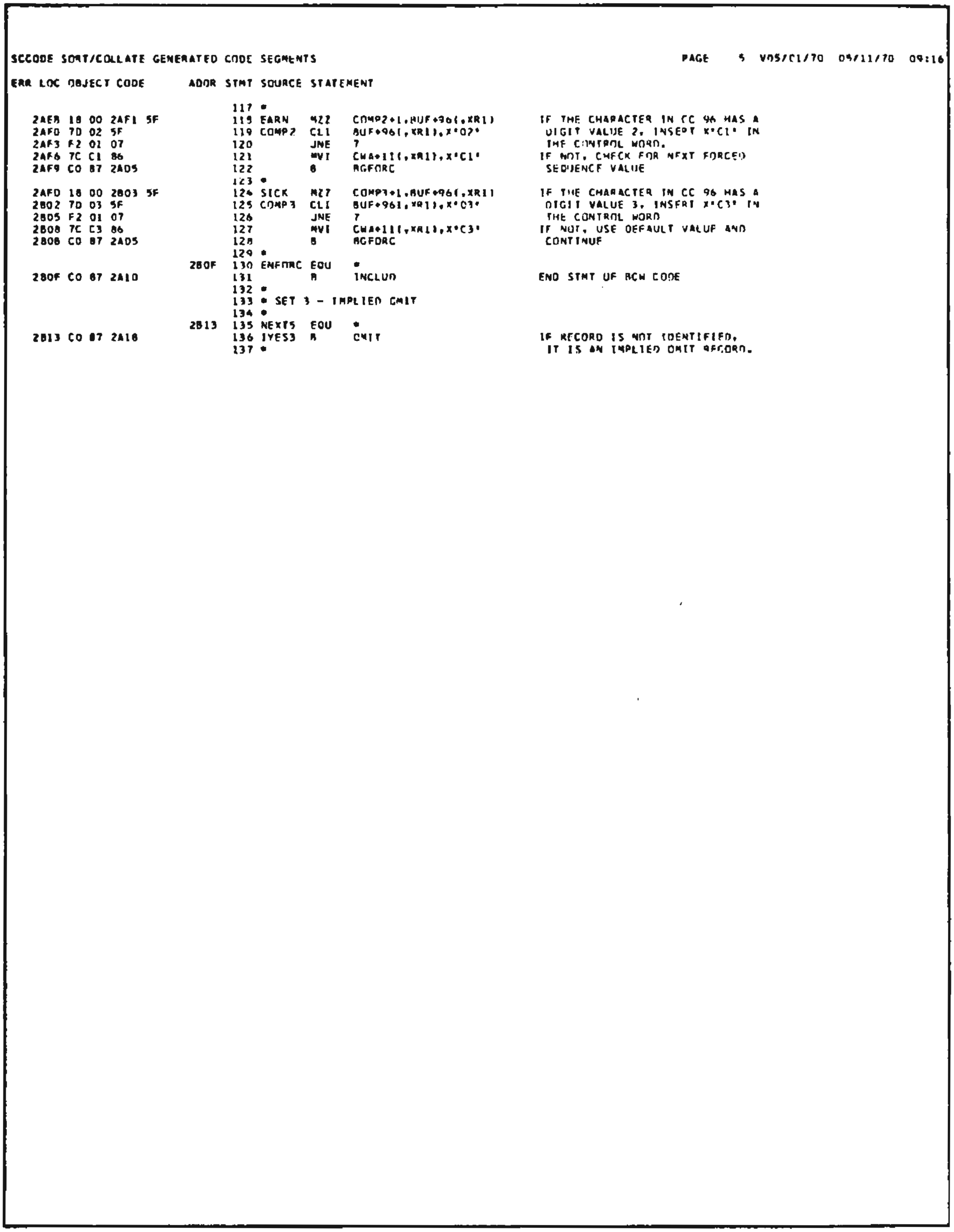
```
SCCODE SORT/COLLATE GENERATED CODE SEGMENTS                                                   PAGE    3  V05/01/70  05/11/70  09:16

ERR LOC OBJECT CODE       ADDR STMT SOURCE STATEMENT

                                17 * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * *
                                18 *                                                                         *
                                19 * GENERATED CODE SEGMENTS                                                 *
                                20 *                                                                         *
                                21 * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * *
                                22 *
                                23 * THE FOLLOWING FOUR INSTRUCTIONS ARE ALWAYS PRESENT
    2A10 3B 80 1BF5             24 INCLUD SBF    SWITCH,INCSW
    2A14 C0 87 1BE9             25        B      JOBMOD
    2A18 3A 80 1BF5             26 OMIT   SBN    SWITCH,OMTSW
    2A1C C0 87 1BE9             27        B      JOBMOD
                                28 *
                                29 * RECORD IDENTIFICATION CODE
                                30 *
                                31 * SET 1
                                32 *
    2A20 C0 87 2A18             33 IYES1  B      OMIT                          IF IN SET 1 - OMIT RECORD
    2A24 C0 87 2A44             34 INO1   B      NEXT2                         LEADING STMT. OF SET 1
                                35 *
    2A28 F2 87 04               36        J      CONST1+1                      CONDITIONS OF RECORD IDENT:
    2A28 F0F7F5F1          2A2E 37 CONST1 DC     CL4'0751'                      1.) EQUAL TEST
    2A2F 4D 03 04 2A2E          38        CLC    BUF+5(4,XR1),CONST1            2.) CHARACTER TEST, LENGTH 4
    2A34 C0 01 2A24             39        BNE    INO1                               IN CC.2 - 5
    2A38 C0 87 2A20             40        B      IYES1                         END STMT. OF SET 1
                                41 *
                                42 * SET 2 - SUBSET A
                                43 *
    2A3C C0 87 2AC4             44 IYES2  B      BCW                           IF IN SET 2 - INCLUDE RECORD
    2A40 C0 87 2A80             45 INO2A  B      NEXT3                           WITH A CONTROL WORD TO BUILD
                                46 *
                           2A44 47 NEXT2  EQU    *
    2A44 F2 87 01               48        J      CONST2+1                      CONDITIONS OF RECORD IDENT:
    2A47 F1                2A47 49 CONST2 DC     CL1'1'                         1.) EQUAL COMPARE
    2A48 54 00 E0 5F            50        ZAZ    WKA+1(1,XR1),BUF+96(1,XR1) 2.) DIGIT TEST, LENGTH 1 IN
    2A4C 7A F0 E0               51        SBN    WKA+1(,XR1),X'F0'                  CC. 96. COMPARE IS DONE
    2A4F 4D 0D E0 2A47          52        CLC    WKA+1(1,XR1),CONST2                IN WORK AREA
    2A54 C0 01 2A40             53        BNE    INO2A
                                54 *
    2A58 F2 87 04               55        J      CONST3+1                      CONDITIONS OF RECORD IDENT.:
    2A5B F0F4F0F0          2A5E 56 CONST3 DC     CL4'0400'                      1.) GREATER OR EQUAL COMPARE
    2A5F 4D 03 04 2A5E          57        CLC    BUF+5(4,XR1),CONST3            2.) CHARACTER TEST, LENGTH 4 IN
    2A64 C0 82 2A40             58        BL     INO2A                              CC 2 - 5.
                                59 *
    2A68 F2 87 04               60        J      CONST4+1                      CONDITIONS OF RECORD IDENT.:
    2A6B F0F5F0F0          2A6E 61 CONST4 DC     CL4'0500'                      1.) LESS THAN COMPARE
    2A6F 4D 03 04 2A6E          62        CLC    BUF+5(4,XR1),CONST4            2.) CHARACTER TEST, LENGTH 4 IN
    2A74 C0 02 2A40             63        BNL    INO2A                              CC 2 - 5.
    2A78 C0 87 2A3C             64        B      IYES2                         END STMT. OF SET 2, SUBSET A
                                65 *
                                66 * SET 2 - SUBSET B
```

Figure 3-13. Symbolic Representation of Generated Code (Part 1 of 3)



```
SCCODE SORT/COLLATE GENERATED CODE SEGMENTS                                                  PAGE    4  V05/01/70  05/11/70  09:16

ERR LOC OBJECT CODE      ADDR STMT SOURCE STATEMENT

                                67 *
   2A7C C0 87 2AAC              68 IN02B  B     NEXT4                     LEADING STMT. OF SET 2, SUBSET B
                                69 *
                         2A80   70 NEXT3  EQU   *
   2A80 F2 87 01                71        J     CONST5+1                  CONDITIONS OF RECORD IDENT.:
   2A83 F2               2A83   72 CONST5 DC    CL1'2'                     1.) EQUAL COMPARE
   2A84 54 00 E0 5F             73        ZAZ   WKA+1(1,XR1),BUF+96(1,XR1) 2.) DIGIT TEST, LENGTH 1 IN
   2A88 7A F0 E0                74        SBN   WKA+1(,XR1),X'F0'              CC. 96. COMPARE IS DONE IN
   2A8B 4D 00 E0 2A83           75        CLC   WKA+1(1,XR1),CONST5            WORK AREA.
   2A90 C0 01 2A7C              76        BNE   IN02B
                                77 *
   2A94 F2 87 04                78        J     CONST6+1                  CONDITIONS OF RECORD IDENT.
   2A97 FDF5F0F0         2A9A   79 CONST6 DC    CL4'0500'                  1.) LESS THAN COMPARE
   2A9B 4D 03 04 2A9A           80        CLC   BUF+5(4,XR1),CONST6        2.) CHARACTER TEST, LENGTH 4 IN
   2AA0 C0 02 2A7C              81        BNL   IN02B                          CC 2 - 5.
   2AA4 C0 87 2A3C              82        B     IYES2                     END STMT. OF SET 2, SUBSET B
                                83 *
                                84 * SET 2 - SUBSET C
                                85 *
   2AA8 C0 87 2B13              86 IN02C  B     NEXT5                     LEADING STMT. OF SET 2, SUBSET C
                                87 *
                         2AAC   88 NEXT4  EQU   *
   2AAC F2 87 01                89        J     CONST7+1                  CONDITIONS OF RECORD IDENT.
   2AAF F3               2AAF   90 CONST7 DC    CL1'3'                     1.) EQUAL COMPARE
   2AB0 54 00 E0 5F             91        ZAZ   WKA+1(1,XR1),BUF+96(1,XR1) 2.) DIGIT TEST, LENGTH 1 IN
   2AB4 7A F0 E0                92        SBN   WKA+1(,XR1),X'F0'              CC. 96. COMPARE IS DONE IN
   2AB7 4D 00 E0 2AAF           93        CLC   WKA+1(,XR1),CONST7             WORK AREA.
   2ABC C0 01 2AA8              94        BNE   IN02C
   2AC0 C0 87 2A3C              95        B     IYES2
                                96 *
                                97 * BCW - CONTROL WORD BUILD CODE
                                98 *
                         2AC4   99 BCW    EQU   *
                               100 *
   2AC4 5C 03 7F 04            101 DEPT   MVC   CWA+4(4,XR1),BUF+5(,XR1)  CHARACTER FIELD
                               102 *
   2AC8 54 05 85 0A            103 MANNO  ZAZ   CWA+10(6,XR1),BUF+11(6,XR1) DIGIT FIELD (ZONES FORCED TO
   2ACC 7A F0 85               104        SBN   CWA+10(,XR1),X'F0'          F'S)
                               105 *
                               106 * FORCED SEQUENCE CODE HAS THREE LEADING INSTRUCTIONS
                               107 *
   2ACF 7C FF 86               108        MVI   CWA+11(,XR1),X'FF'        INSERTS DEFAULT VALUE
   2AD2 F2 87 04               109        J     4
   2AD5 C0 87 2B0F             110 BGFORC B     ENFORC                    EXIT FROM FORCE SEQ. TO NEXT COD
                               111 *
   2AD9 18 00 2ADF 5F          112 DEDUCT MZZ   COMP1+1,BUF+96(,XR1)      IF THE CHARACTER IN CC 96 HAS A
   2ADE 7D 01 5F               113 COMP1  CLI   BUF+96(,XR1),X'01'         DIGIT VALUE 1, INSERT X'C2'
   2AE1 F2 01 07               114        JNE   7                          IN THE CONTROL WORD
   2AE4 7C C2 86               115        MVI   CWA+11(,XR1),X'C2'        IF NOT, CHECK FOR NEXT FORCED
   2AE7 C0 87 2AD5             116        B     BGFORC                     SEQUENCE VALUE
```

Figure 3-13. Symbolic Representation of Generated Code (Part 2 of 3)



```
SCCODE SORT/COLLATE GENERATED CODE SEGMENTS                                                   PAGE    5  V05/C1/70  09/11/70  09:16

ERR LOC OBJECT CODE      ADDR STMT SOURCE STATEMENT

                               117 *
    2AEB 18 00 2AF1 5F         118 EARN   MZZ    COMP2+1,BUF+96(,XR1)        IF THE CHARACTER IN CC 96 HAS A
    2AF0 7D 02 5F              119 COMP2  CLI    BUF+96(,XR1),X'02'           DIGIT VALUE 2, INSERT X'C1' IN
    2AF3 F2 01 07              120        JNE    7                            THE CONTROL WORD.
    2AF6 7C C1 86              121        MVI    CWA+11(,XR1),X'C1'          IF NOT, CHECK FOR NEXT FORCED
    2AF9 C0 87 2AD5            122        B      RGFORC                       SEQUENCE VALUE
                               123 *
    2AFD 18 00 2B03 5F         124 SICK   MZZ    COMP3+1,BUF+96(,XR1)        IF THE CHARACTER IN CC 96 HAS A
    2B02 7D 03 5F              125 COMP3  CLI    BUF+961,XR1),X'03'           DIGIT VALUE 3, INSERT X'C3' IN
    2B05 F2 01 07              126        JNE    7                            THE CONTROL WORD
    2B08 7C C3 86              127        MVI    CWA+11(,XR1),X'C3'          IF NOT, USE DEFAULT VALUE AND
    2B0B C0 87 2AD5            128        B      RGFORC                       CONTINUE
                               129 *
                         2B0F  130 ENFORC EQU    *
    2B0F C0 87 2A10            131        B      INCLUD                      END STMT OF RCW CODE
                               132 *
                               133 * SET 3 - IMPLIED OMIT
                               134 *
                         2B13  135 NEXT5  EQU    *
    2B13 C0 87 2A18            136 IYES3  B      OMIT                        IF RECORD IS NOT IDENTIFIED,
                               137 *                                          IT IS AN IMPLIED OMIT RECORD.
```

Figure 3-13. Symbolic Representation of Generated Code (Part 3 of 3)

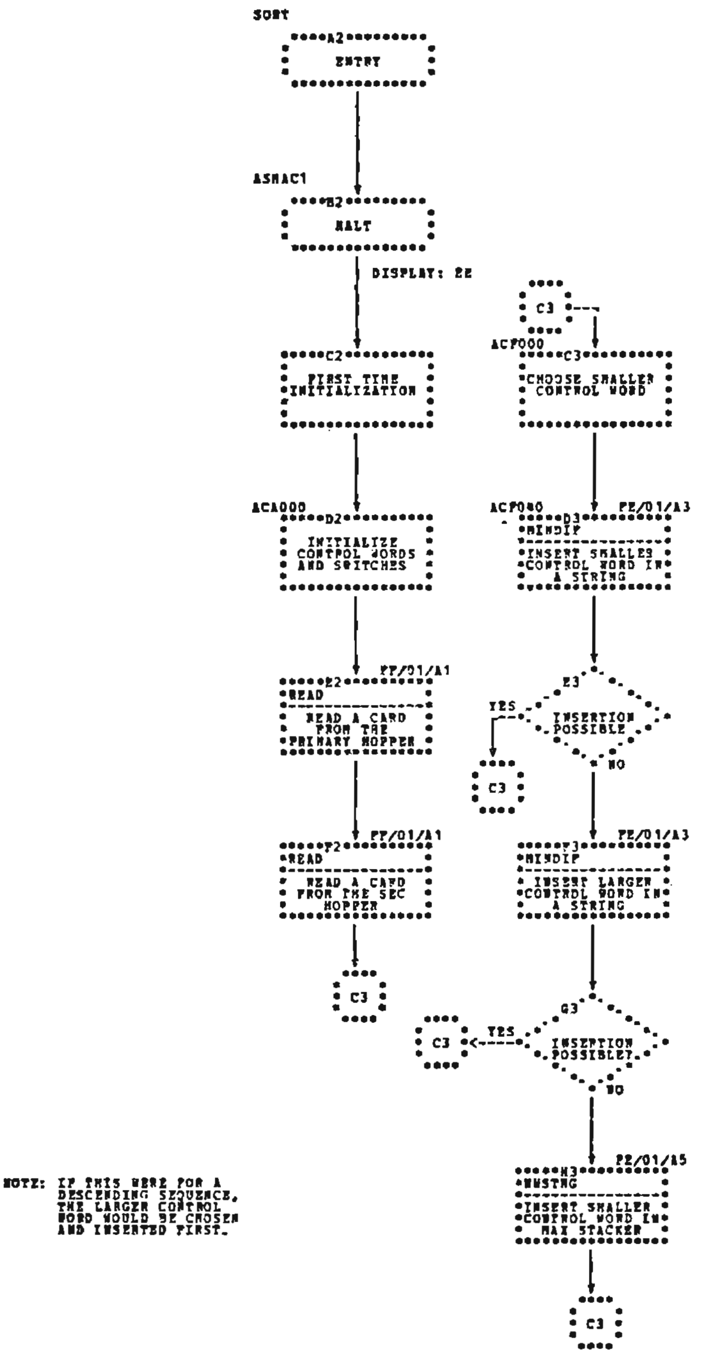

SORT
A2
ENTRY
ASMAC1
B2
HALT
DISPLAY: EE
C3
ACF000
C2
FIRST TIME INITIALIZATION
C3
CHOOSE SMALLER CONTROL WORD
ACA000
D2
INITIALIZE CONTROL WORDS AND SWITCHES
ACF080
D3
FE/01/A3
MINDIF
INSERT SMALLER CONTROL WORD IN A STRING
FF/01/A1
E2
READ
READ A CARD FROM THE PRIMARY HOPPER
E3
YES
INSERTION POSSIBLE
NO
C3
FF/01/A1
F2
READ
READ A CARD FROM THE SEC HOPPER
C3
F3
FE/01/A3
MINDIF
INSERT LARGER CONTROL WORD IN A STRING
G3
C3
YES
INSERTION POSSIBLE?
NO
H3
FE/01/A5
MNSTNG
INSERT SMALLER CONTROL WORD IN MAX STACKER
C3
NOTE: IF THIS WERE FOR A DESCENDING SEQUENCE, THE LARGER CONTROL WORD WOULD BE CHOSEN AND INSERTED FIRST.

Chart FA. Sort Job Module ($CSSRT)

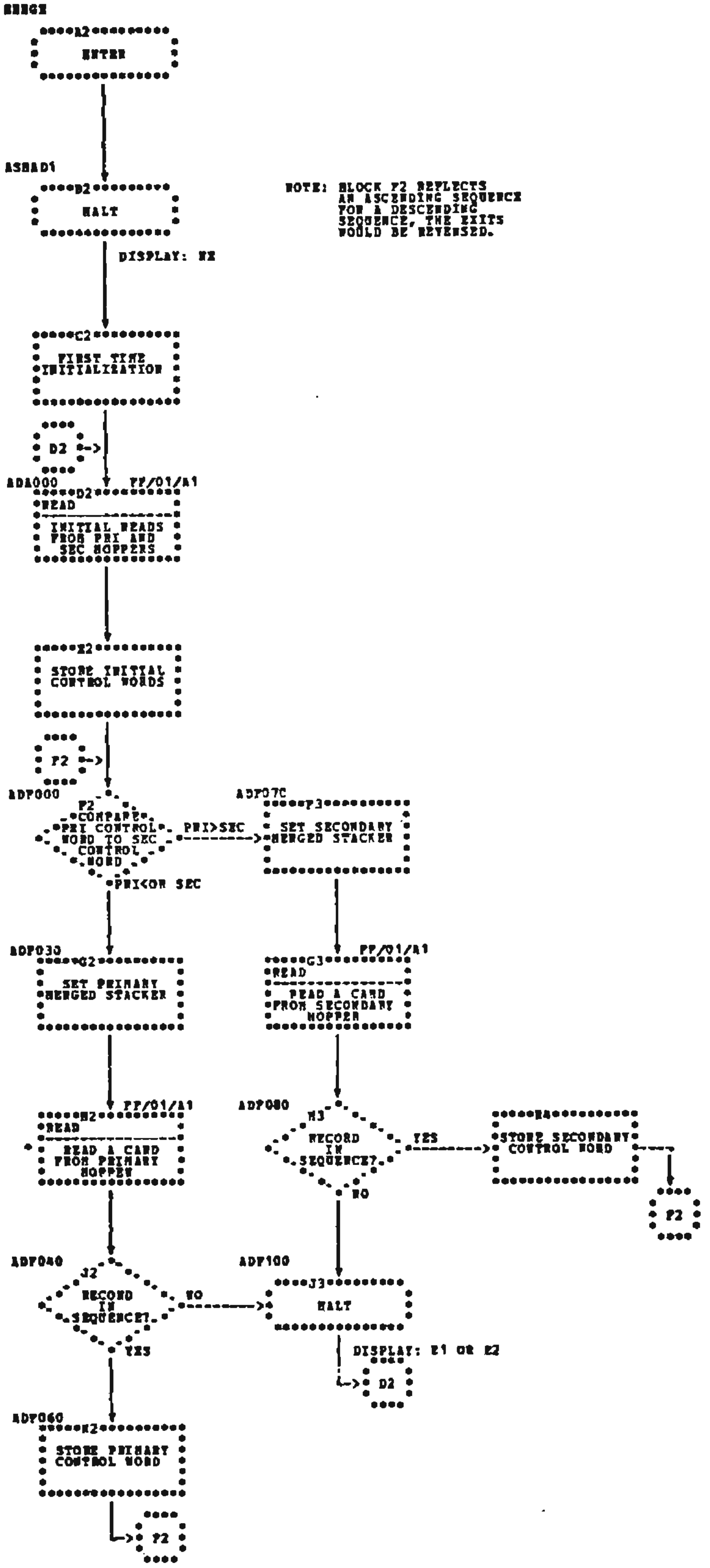

MERGE
A2
ENTER
ASMAD1
B2
HALT
DISPLAY: MZ
NOTE: BLOCK F2 REFLECTS AN ASCENDING SEQUENCE FOR A DESCENDING SEQUENCE, THE EXITS WOULD BE REVERSED.
C2
FIRST TIME INITIALIZATION
D2
ADA000
FF/01/A1
D2
READ
INITIAL READS FROM PRI AND SEC HOPPERS
E2
STORE INITIAL CONTROL WORDS
F2
ADF000
F2
COMPARE PRI CONTROL WORD TO SEC CONTROL WORD
PRI>SEC
PRI<OR SEC
ADF07C
F3
SET SECONDARY MERGED STACKER
ADF030
G2
SET PRIMARY MERGED STACKER
FF/01/A1
G3
READ
READ A CARD FROM SECONDARY HOPPER
FF/01/A1
H2
READ
READ A CARD FROM PRIMARY HOPPER
ADF080
H3
RECORD IN SEQUENCE?
YES
NO
H4
STORE SECONDARY CONTROL WORD
F2
ADF040
J2
RECORD IN SEQUENCE?
NO
YES
ADF100
J3
HALT
DISPLAY: E1 OR E2
D2
ADF060
K2
STORE PRIMARY CONTROL WORD
F2

Chart FB. Merge Job Module ($CSMRG)

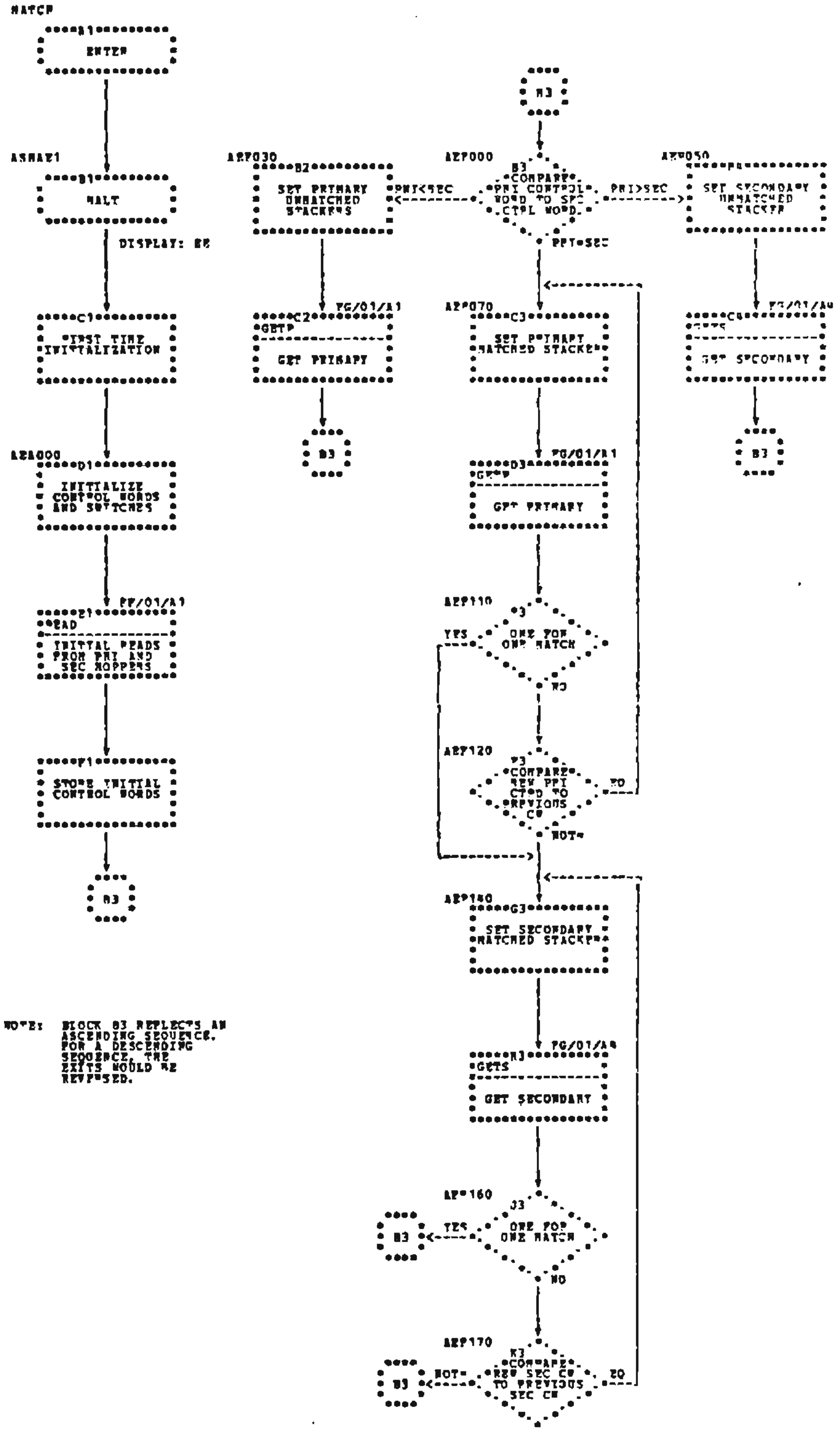

MATCH
A1
ENTER
ASHAE1
B1
HALT
DISPLAY: EE
C1
FIRST TIME INITIALIZATION
AEA000
D1
INITIALIZE CONTROL WORDS AND SWITCHES
FF/01/A1
E1
READ
INITIAL READS FROM PRI AND SEC HOPPERS
F1
STORE INITIAL CONTROL WORDS
B3
AEF030
B2
SET PRIMARY UNMATCHED STACKERS
PRI<SEC
FG/01/A1
C2
GETP
GET PRIMARY
B3
B3
AEF000
B3
COMPARE PRI CONTROL WORD TO SEC CTRL WORD
PRI>SEC
PRI=SEC
AEF050
B4
SET SECONDARY UNMATCHED STACKER
FG/01/A4
C4
GETS
GET SECONDARY
B3
AEF070
C3
SET PRIMARY MATCHED STACKERS
FG/01/A1
D3
GETP
GET PRIMARY
AEF110
E3
YES
ONE FOR ONE MATCH
NO
AEF120
F3
COMPARE NEW PRI CTRL TO PREVIOUS CW
EQ
NOT=
AEF140
G3
SET SECONDARY MATCHED STACKERS
FG/01/A4
H3
GETS
GET SECONDARY
AEF160
J3
B3
YES
ONE FOR ONE MATCH
NO
AEF170
K3
B3
NOT=
COMPARE NEW SEC CW TO PREVIOUS SEC CW
EQ
NOTE: BLOCK B3 REFLECTS AN ASCENDING SEQUENCE. FOR A DESCENDING SEQUENCE, THE EXITS WOULD BE REVERSED.

Chart FC. Match Job Module ($CSMCH)

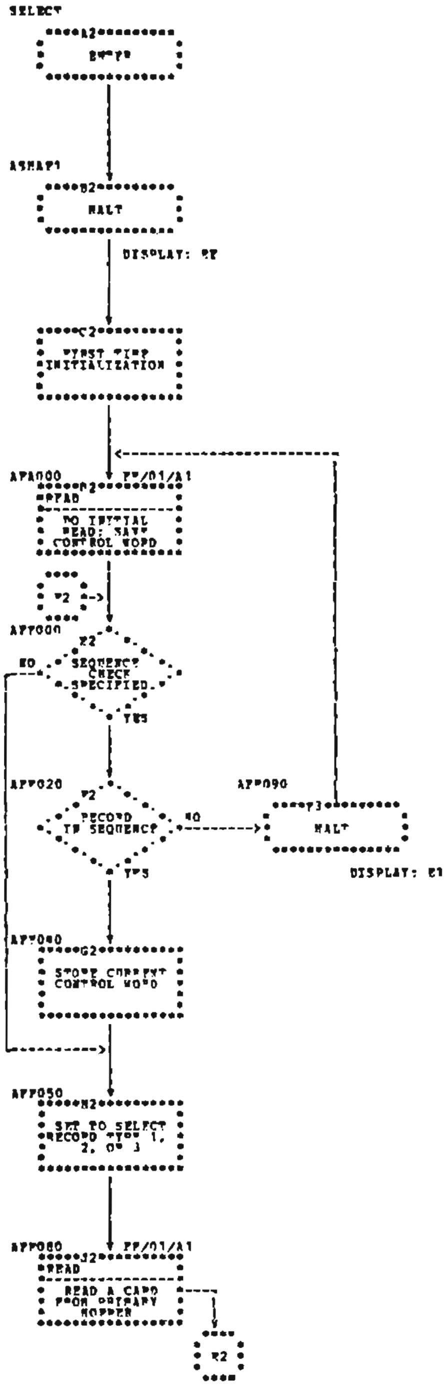
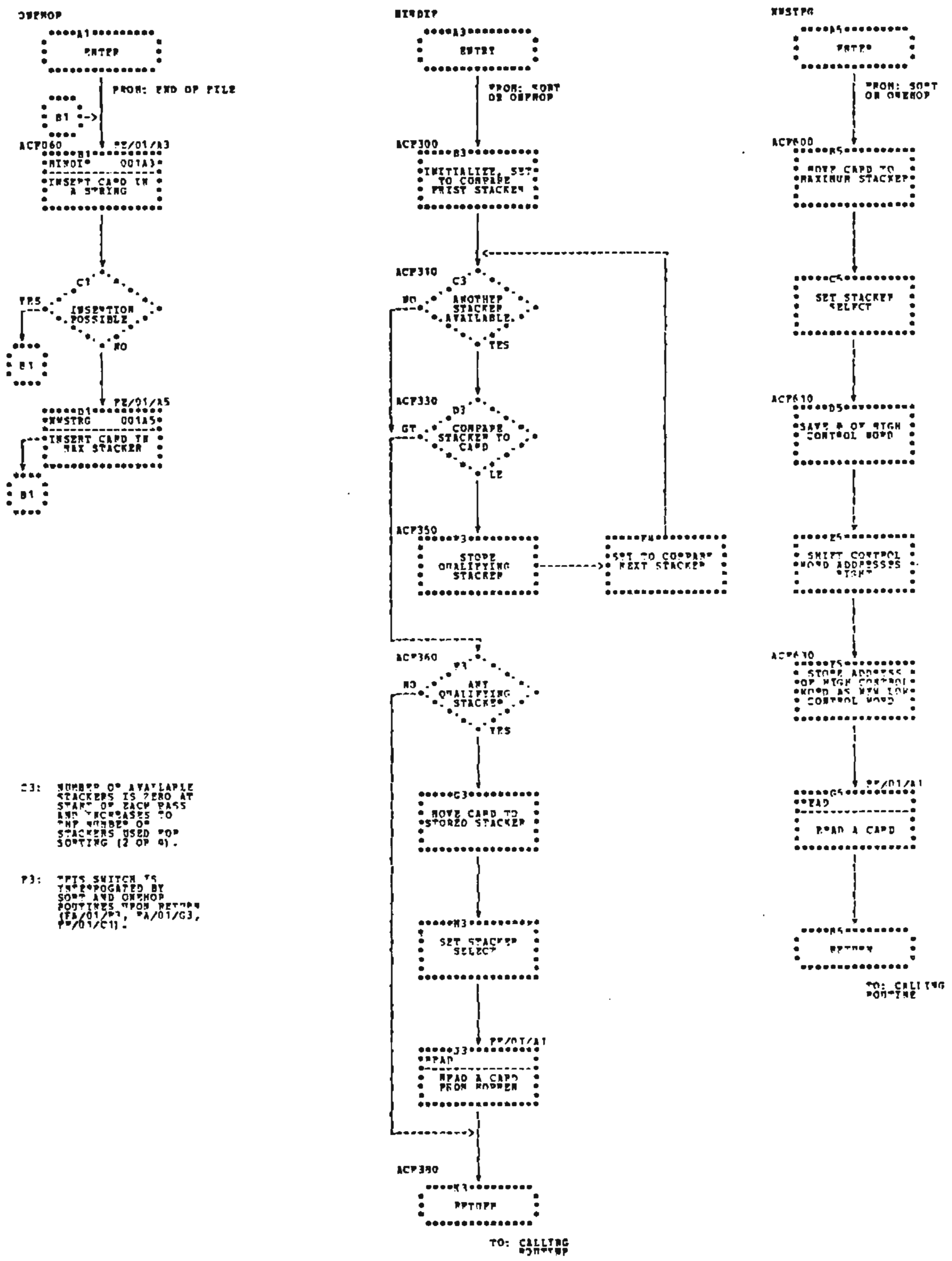


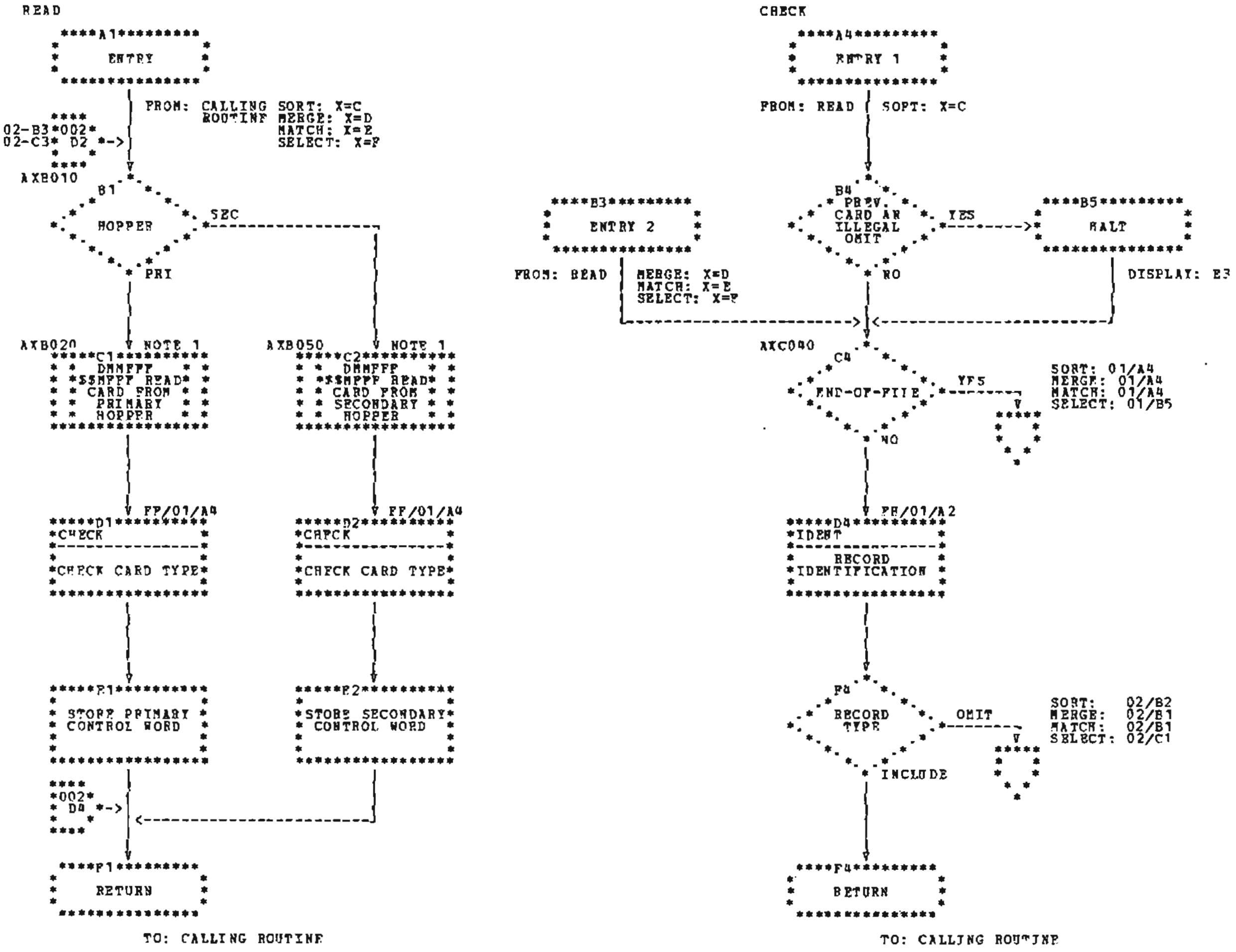
Chart FD. Select Job Module ($CSSEL)



C3: NUMBER OF AVAILABLE STACKERS IS ZERO AT START OF EACH PASS AND INCREASES TO THE NUMBER OF STACKERS USED FOR SORTING (2 OR 4).

F3: THIS SWITCH IS INTERROGATED BY SORT AND ONEHOP ROUTINES UPON RETURN (FA/01/F3, FA/01/G3, FB/01/C1).

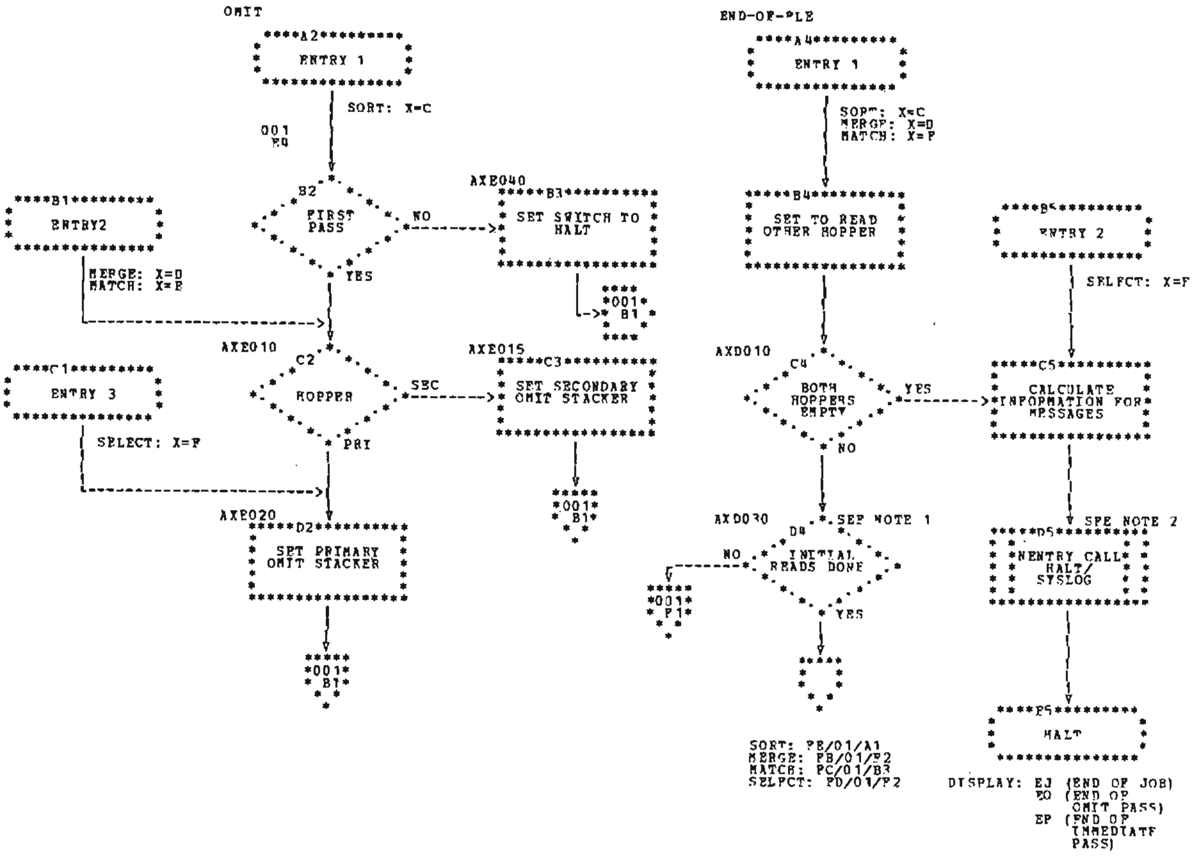
Chart FE. Routines Used Sort Job Module



NOTE 1: THE $$MFFF ROUTINE IS FOUND IN IBM SYSTEM/3 DISK SYSTEMS DATA MANAGEMENT AND INPUT/OUTPUT SUPERVISOR LOGIC MANUAL SY21-0512.

NOTE 2: THE INSTRUCTIONS AT LABELS AXB040 AND AXB070 EXIST ONLY IN THE SORT JOB MODULE

Chart FF. Common Routines for All Job Modules (Part 1 of 2)



NOTE 1: IF NO INCLUDE CARDS HAVE BEEN READ, RETURN IS TO ADDRESS SAVED IN THE READ ROUTINE

NOTE 2: THE HALT/SYSLOG ROUTINE IS FOUND IN IBM SYSTEM/3 DISK SYSTEMS SYSTEM CONTROL PROGRAM LOGIC MANUAL, SY21-0502

Chart FF. Common Routines for All Job Modules (Part 2 of 2)

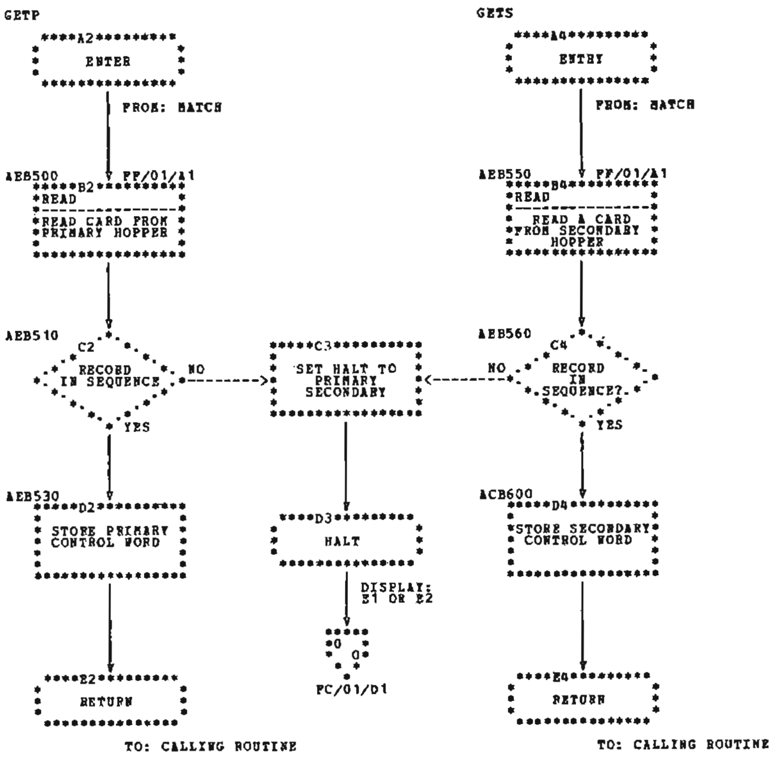
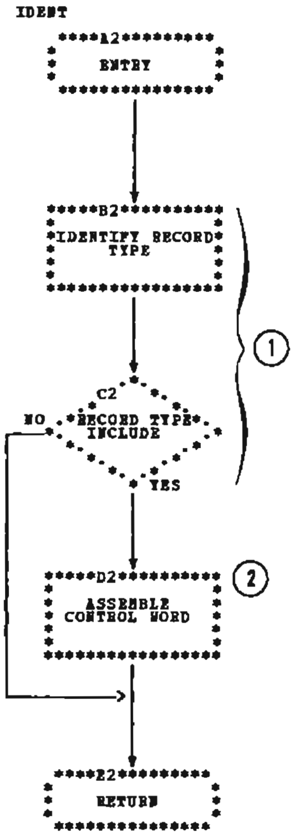


Chart FG. Common Routines for Merge and Match Job Modules



①

Generated code segments for I and O type specifications:
Beginning of a Set
Beginning of a Subset
Set Stacker for Stacker Select
Jump Over Constant
Character—Field to Constant
Character—Field to Field
Zone
Digit—Field to Constant
Digit—Field to Field
Unpacked—Field to Constant
Unpacked—Field to Field
Branch on Condition Instruction
Branch Instruction

②

Generated code segments for F type specifications:
Jump Over Constant
Normal Field—Character
Normal Field—Zone
Normal Field—Digit
Opposite Field—Digit
Unpacked Field—Normal or Opposite
Force Sequence—Leading Instruction
Beginning of Force Lines
Forced Field—Character (Part 1)
Forced Field—Zone (Part 1)
Forced Field—Digit (Part 1)
Force-All
Forced Field—Character, Zone, Digit (Part 2)
Branch to Include

Chart FH. Record Identification Routine (Generated Code Segments)

Part IV. Gangpunch

Section 1. Introduction

The Gangpunch program is a disk resident program which provides the following three types of gangpunching:

- Interspersed gangpunching. Master and detail cards are intermixed in the primary file. The detail records are punched and interpreted according to the header and field definition specifications.

- Count-controlled gangpunching. Detail cards are in the primary file and master cards are in the secondary file. Either a constant or variable counter can be used to punch and interpret a specified number of detail cards according to the header and field definition specifications.

- Match-field gangpunching. Detail cards are in the primary file and master cards are in the secondary file. Match fields are defined on the detail and master cards. When identical detail and master card match fields are found, the detail card is punched and interpreted according to the header and field definition specifications.

The following functions are also provided for any of the three previous gangpunching types:

- Offset gangpunching

- Gangpunching consecutive numbers into detail cards

- Gangpunching a constant into detail cards

- Interpreting detail cards (either the entire card or only the data that has been punched in the card)

- Selecting a single type of master card from many master cards

- Selecting a single type of detail card from many detail cards

SYSTEM REQUIREMENTS

The Gangpunch program requires:

- IBM 5410 Processing Unit Model A13 (12K)

- IBM 5203 or 1403 Printer

- IBM 5424 MFCU

- IBM 5444 Disk Storage Drive

PROGRAM STRUCTURE

The Gangpunch program consists of two phases – the diagnostic and execution phases.

1. The diagnostic phase:

   - Reads and diagnoses the header record

   - Reads and diagnoses the field definition record(s)

   - Builds an FDP table and a common area that are used by the execution phase

   - Prints all error messages

   - Cancels the job if terminal errors have occurred

   - Gives control to the execution phase

2. The execution phase gangpunches detail records according to the header and field definition records processed by the diagnostic phase.

Section 2. Method of Operation

This section describes the functions of the Gangpunch program and relates each function to the part (routine) that performs the function. Three types of diagrams are used to describe the functional organization of the Gangpunch program — a visual table of contents, overview diagrams, and lower level diagrams.

The visual table of contents is an overall picture of the program. It enables the reader to skip directly to a particular diagram instead of following the diagram tree structure. Diagram 0 is an example of a visual table of contents.

The overview diagram describes the functions in general. It refers to lower level diagrams. Diagram 1 is an example of an overview diagram.

The lower level diagrams describe the function, the input required, and the output produced. Diagram 2 is an example of a lower level diagram. Each lower level diagram has four major areas:

1. Input: Shows the input that is required to perform the function. It appears in the left column of the diagram.

2. Process: Describes the steps taken to perform the function. The steps appear in the center column of the diagram.

3. Output: Shows the output produced by the function. It appears in the right column of the diagram.

4. Extended Description: Gives cross references to the routines (routine name and label) that perform the function. It appears in the center column of the diagram beneath the process block.

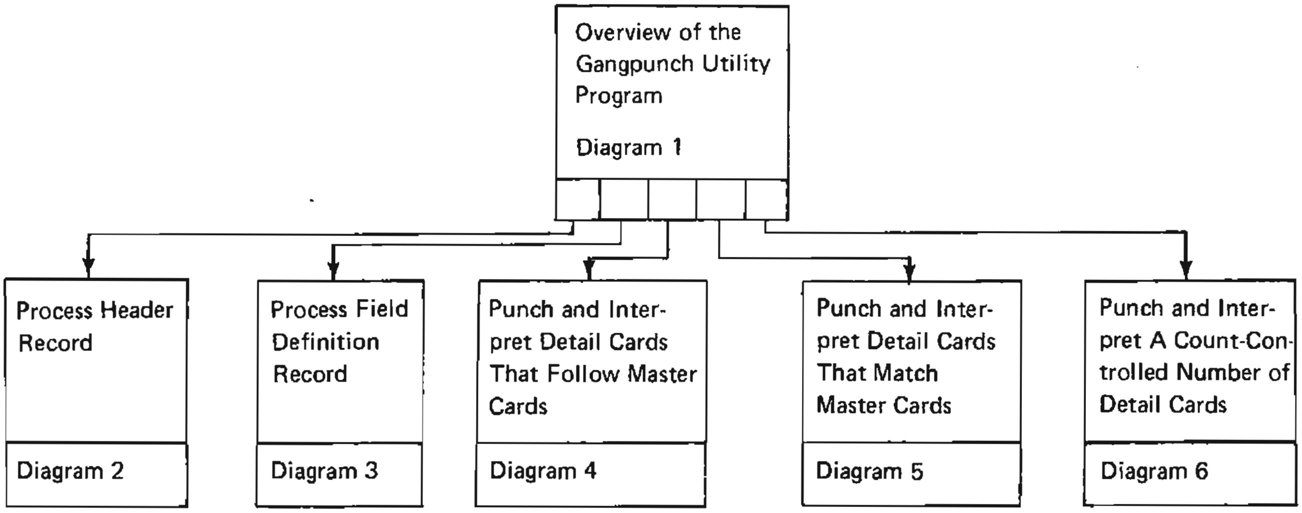
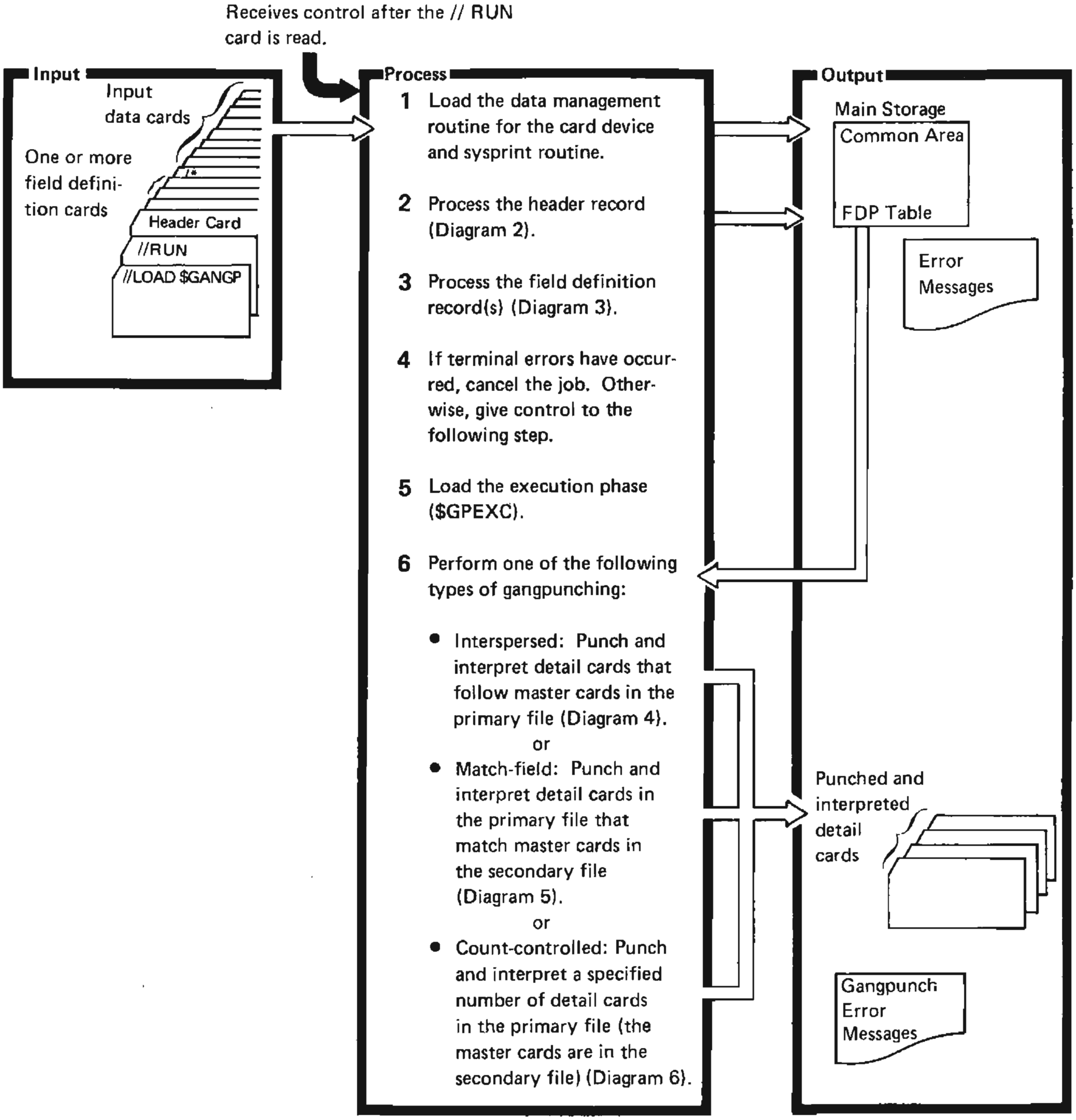

Overview of the Gangpunch Utility Program
Diagram 1
Process Header Record
Diagram 2
Process Field Definition Record
Diagram 3
Punch and Interpret Detail Cards That Follow Master Cards
Diagram 4
Punch and Interpret Detail Cards That Match Master Cards
Diagram 5
Punch and Interpret A Count-Controlled Number of Detail Cards
Diagram 6

Diagram 0. Visual Table of Contents for the Gangpunch Program Documentation



Process

1 Load the data management routine for the card device and sysprint routine.

2 Process the header record (Diagram 2).

3 Process the field definition record(s) (Diagram 3).

4 If terminal errors have occurred, cancel the job. Otherwise, give control to the following step.

5 Load the execution phase ($GPEXC).

6 Perform one of the following types of gangpunching:

- Interspersed: Punch and interpret detail cards that follow master cards in the primary file (Diagram 4).

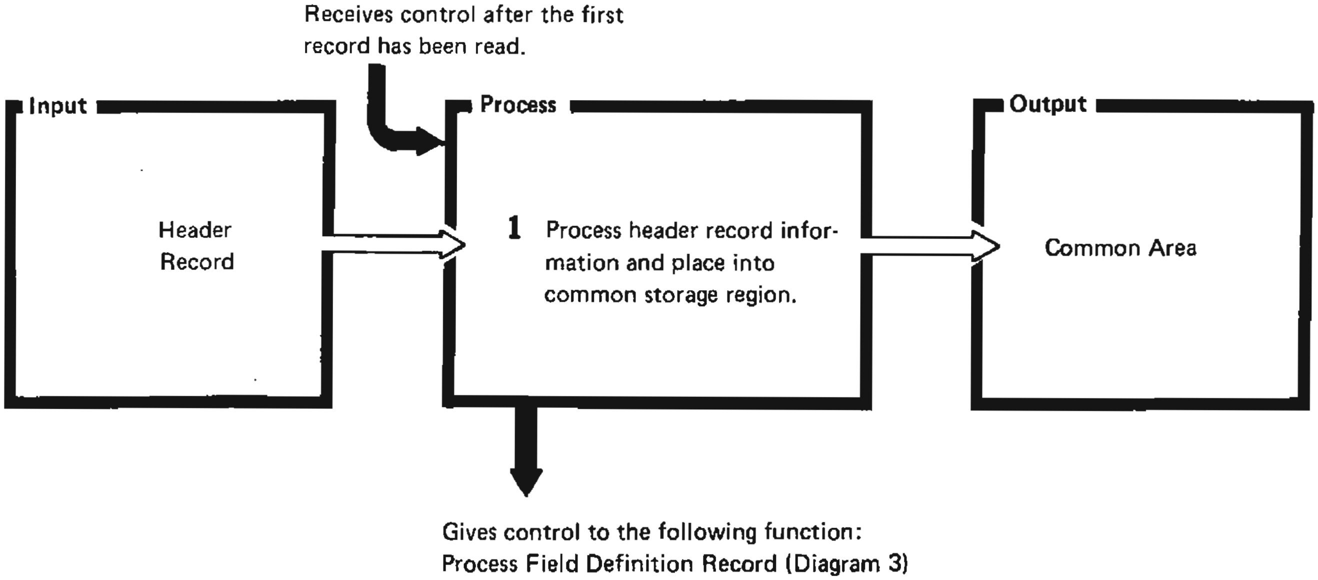
  or

- Match-field: Punch and interpret detail cards in the primary file that match master cards in the secondary file (Diagram 5).

  or

- Count-controlled: Punch and interpret a specified number of detail cards in the primary file (the master cards are in the secondary file) (Diagram 6).

Diagram 1. Overview of the Gangpunch Program Diagnostic and Execution Phases

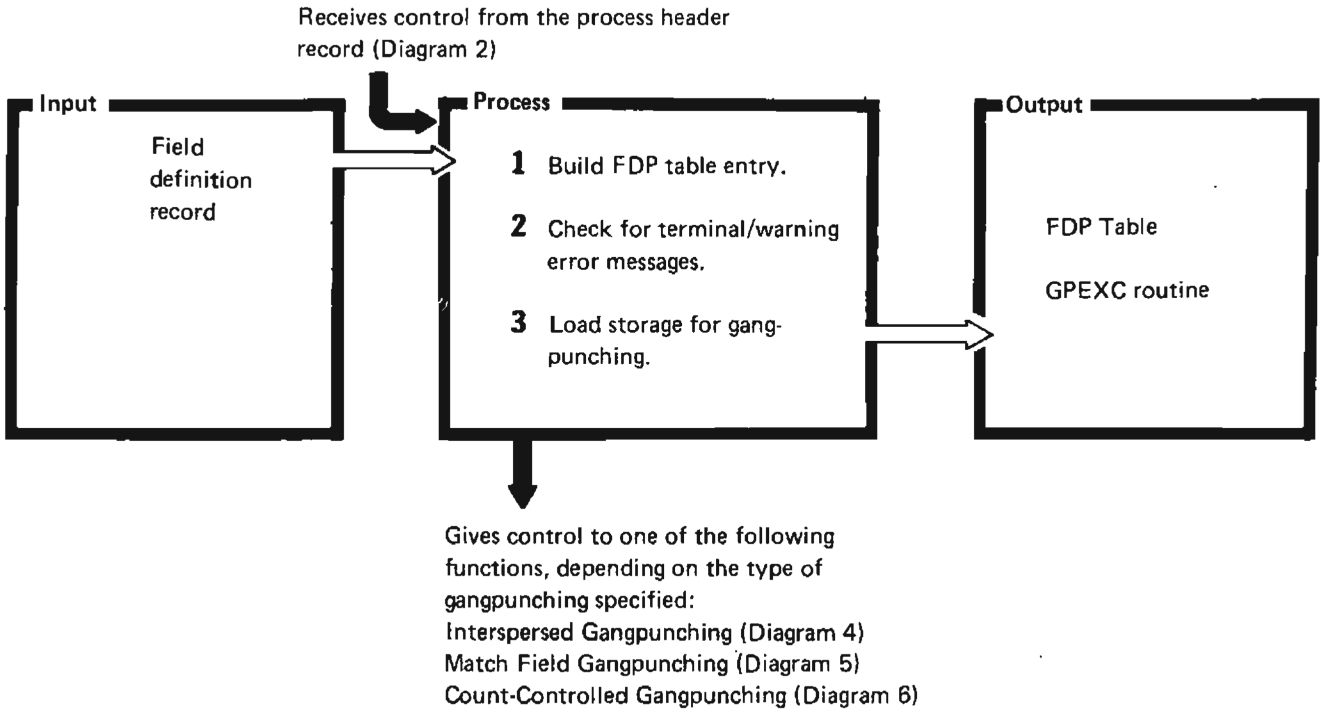

Input
Header Record
Process
1 Process header record information and place into common storage region.
Output
Common Area

Receives control after the first record has been read.

Gives control to the following function: Process Field Definition Record (Diagram 3)

Module	Label
1 $GANGP	CHKCRD

Diagram 2. Process Header Record



Receives control from the process header record (Diagram 2)

Input

Field definition record

Process

1 Build FDP table entry.

2 Check for terminal/warning error messages.

3 Load storage for gangpunching.

Output

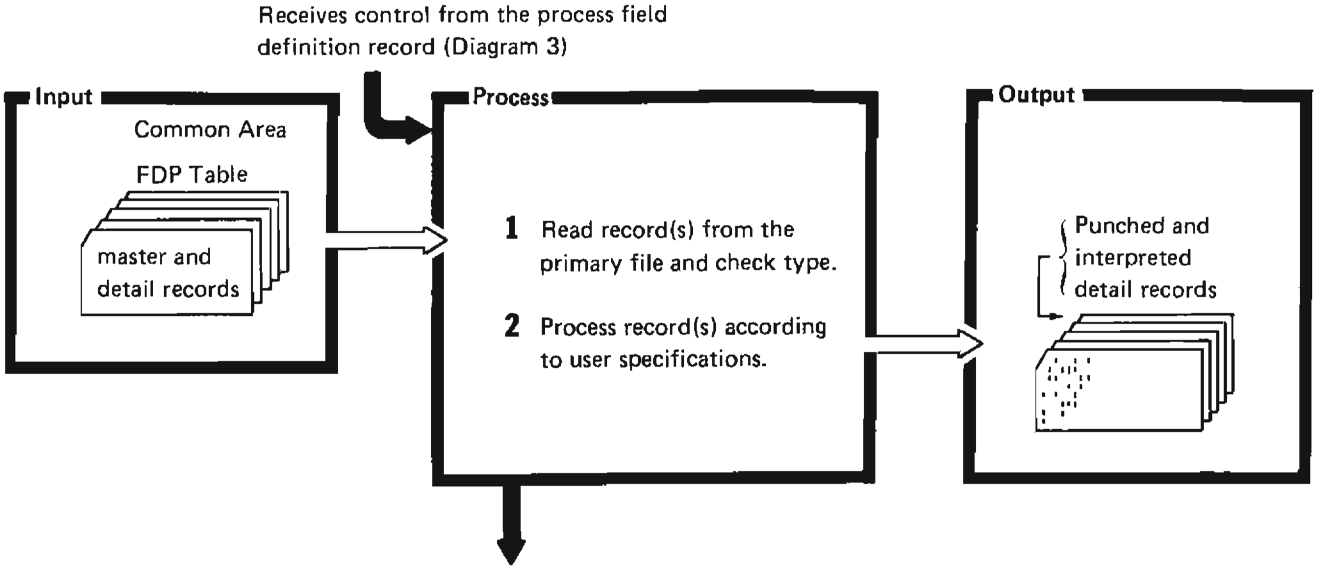
FDP Table

GPEXC routine

Gives control to one of the following functions, depending on the type of gangpunching specified:
Interspersed Gangpunching (Diagram 4)
Match Field Gangpunching (Diagram 5)
Count-Controlled Gangpunching (Diagram 6)

	Module	Label
1	$GANGP	CK07FD
2	$GANGP	ENDTST
3	$GANGP	ENDGP

Diagram 3. Process Field Definition Record



Receives control from the process field definition record (Diagram 3)

Input

Common Area

FDP Table

master and detail records

Process

1 Read record(s) from the primary file and check type.

2 Process record(s) according to user specifications.

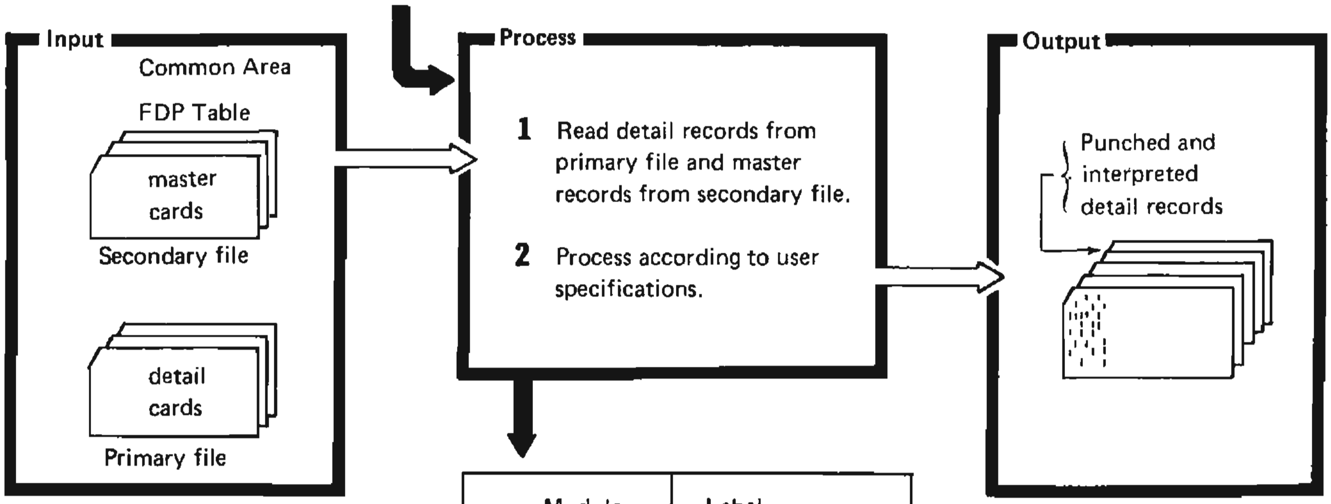
Output

Punched and interpreted detail records

Module	Label
1 $GPEXC	INTPRC
2 $GPEXC	INTPRC

Diagram 4. Interspersed Gangpunching

Receives control from the process field definition record (Diagram 3)

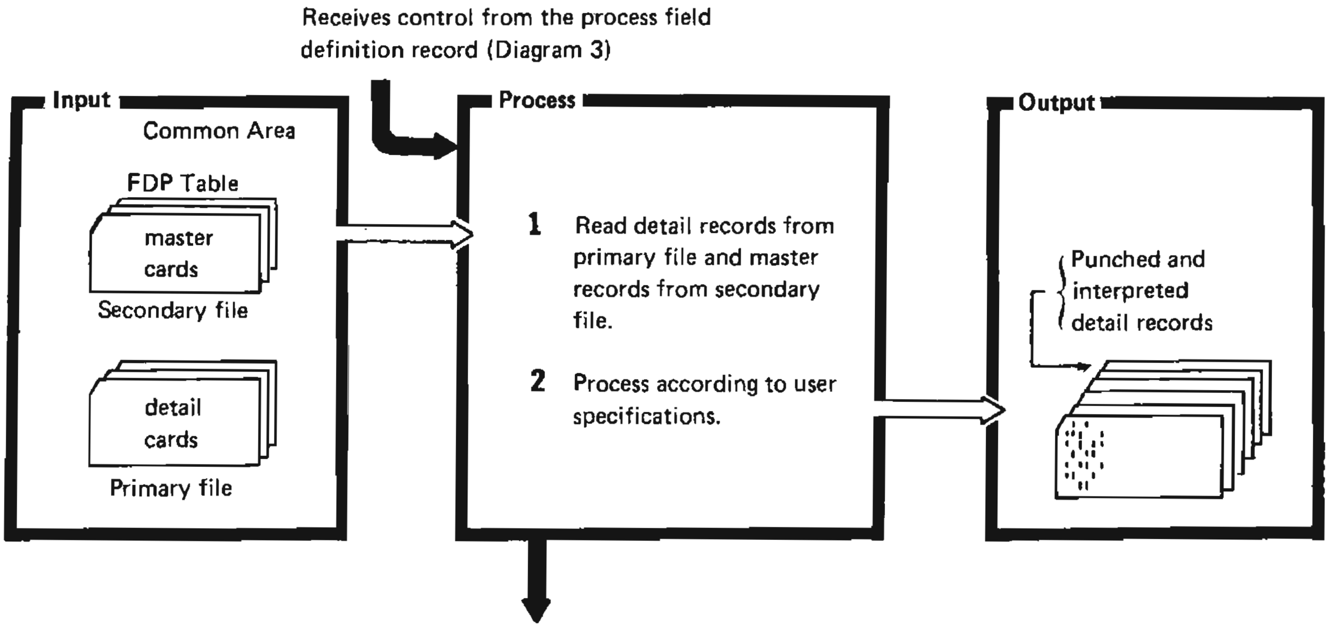


1 Read detail records from primary file and master records from secondary file.

2 Process according to user specifications.

	Module	Label
1	$GPEXC	MATPRC
2	$GPEXC	MATPRC

Diagram 5. Match-Field Gangpunching



Receives control from the process field definition record (Diagram 3)
Input
Common Area
FDP Table
master cards
Secondary file
detail cards
Primary file
Process
Output
Punched and interpreted detail records

1 Read detail records from primary file and master records from secondary file.

2 Process according to user specifications.

	Module	Label
1	$GPEXC	CCPRC
2	$GPEXC	CCPRC

Diagram 6. Count-Controlled Gangpunching

Section 3. Program Organization

PHASE DESCRIPTIONS

This section gives a detailed description of the diagnostic and execution phases of the Gangpunch program. Each phase is explained by listing its entry point, general functions, input, output, and routines called. The routines used by each phase are explained by listing the entry point, functions, input, output, and any routines called.

The diagnostic phase is loaded when the // LOAD $GANGP, UNIT OCL statement is read. The execution phase is loaded by the diagnostic phase after the header record and field definition record(s) have been diagnosed. Figure 4-1 shows a storage map for the phases.

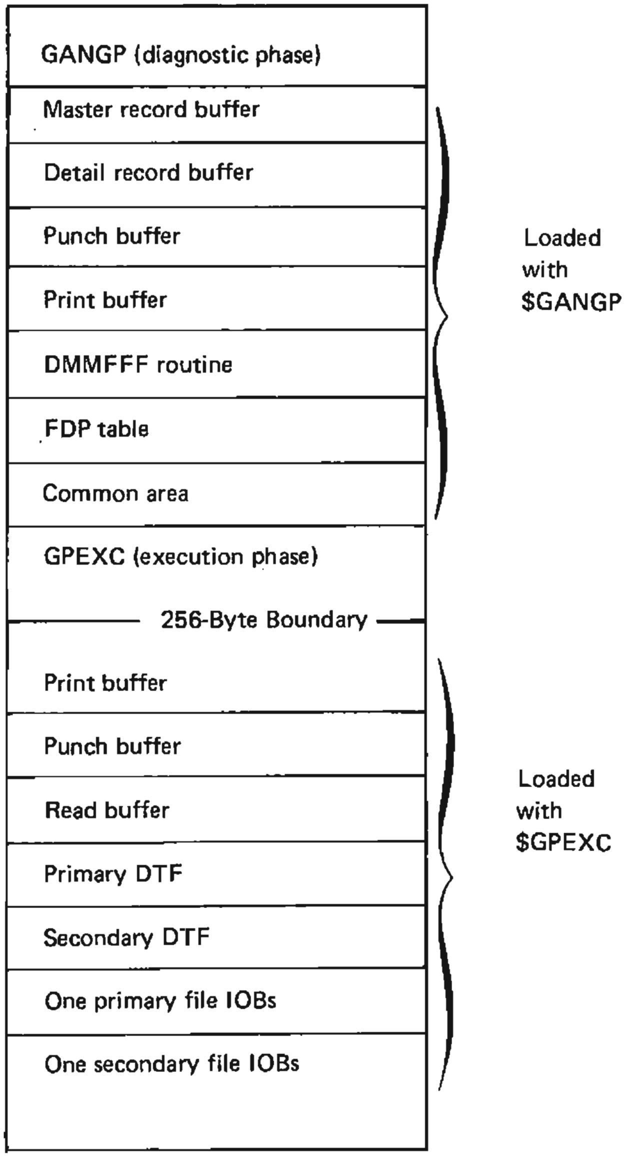


Figure 4-1. Storage Map for the Diagnostic and Execution Phases

Diagnostic Phase (GANGP)

Entry Point: GANGP

Chart: DA

Functions:

- Print the following Gangpunch program heading: SYSTEM/3 MODEL 10 GANGPUNCH VERSION XX MODIFICATION LEVEL XX date from communication area.
- Check that the first record read is the header record.
- Print the header record.
- Process the header record, column by column; build the common area using header record data; if errors are detected, indicate them in ERTAB1.
- Detect invalid decimal digits in columns 9-10, 14-15, 19-20, 24, 25, 29-36, and 53-60 of the header record and print the character S under the previously printed header record to indicate each position in error.
- Print error messages that have been flagged in diagnosing the header record.
- Check for invalid field definition records following the header record.
- Process the field definition record, column by column; build an FDP table entry for each record; if errors are detected, indicate them in ERTAB1.
- Indicate a terminal error if too many field definition records have been read.
- Print all field definition record error messages after each field definition record has been read.
- If terminal errors have been flagged, log the message ERRORS IN SPECIFICATIONS and cancel the job.
- If warning errors have been found, log the message REVIEW WARNING MESSAGES and give the operator the option of cancelling the job.
- Load the execution phase, GPEXC, and give it control.

Input:

- Header record
- Field definition record(s)

Output:

- Printed header record
- Printed field definition record(s)
- Printed error messages
- Common region
- FDP table

Routines Called:

- DMMFFF
- LOAD
- SYSPNT
- FIND

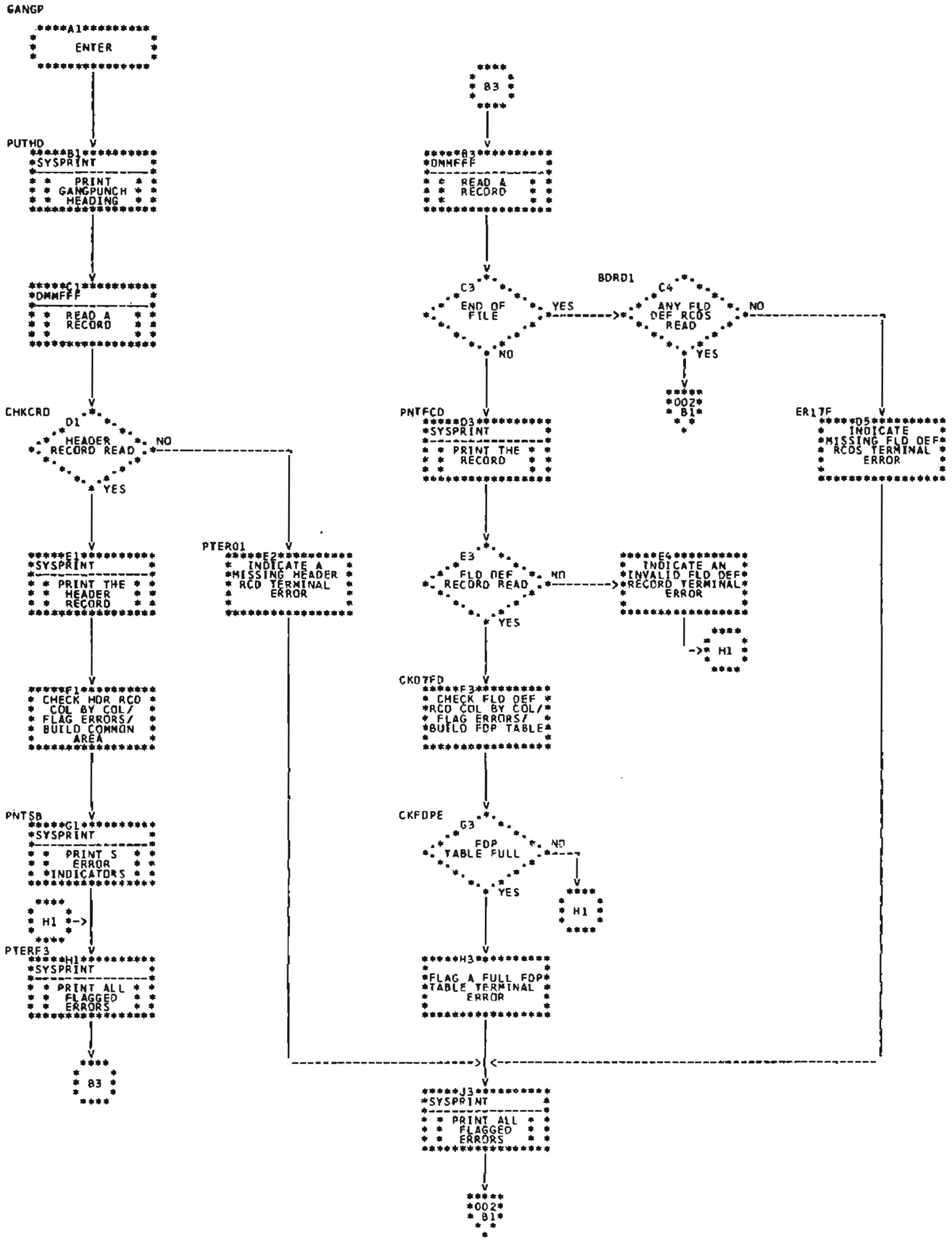


Chart DA (Part 1 of 2). Diagnostic Phase (GANGP)

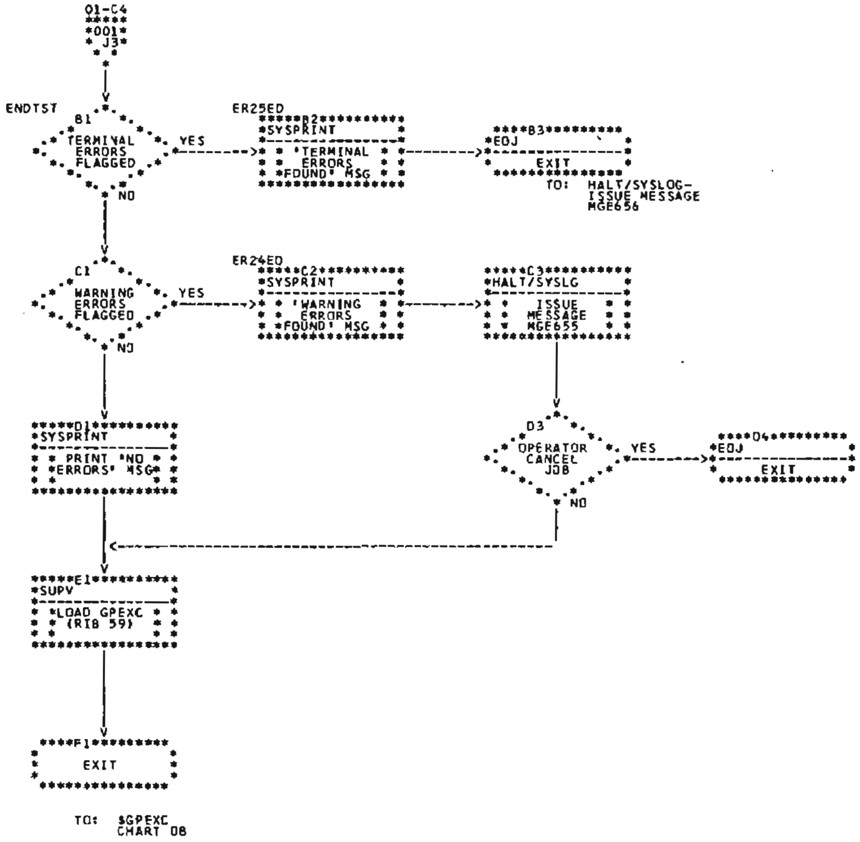


Chart DA (Part 2 of 2). Diagnostic Phase (GANGP)

Decimal to Binary Conversion Routine

Entry Point: CONVRT

Functions:

- Converts the 2-byte decimal field addressed by register 2 to binary.

- Stores the result in the leftmost byte of the same field.

- Indicates an invalid decimal field by placing X'FF' in the leftmost byte of the field.

Input:

- Two-byte decimal field addressed by register 2.

Output:

- One-byte binary field addressed by register 2.

Routines Called: None

Numeric Field Test Routine

Entry Point: CFCONT

Functions:

- Checks that the constant starting and ending values specified for the counter are valid numeric constants.

- Indicates an invalid counter value by flagging error GP12 in ERTAB1.

Input:

- Register 2 addresses the counter value to be tested.

Output:

- A blank counter value is set to zero.

- Error GP12 is flagged in ERTAB1 if the counter value is invalid.

Routines Called: None

Master or Detail Record Selector Information Identification Check Routine

Entry Point: IDCHK

Functions:

- Converts the first two bytes of the selector information to binary using the Decimal to Binary Conversion Routine.

- Flags error GP04 in ERTAB1 if the first two bytes could not be converted.

- Indicates the not condition (if specified) in SELCOD.

- Flags error GP09 if a character other than a blank or N is specified.

- Indicates the type of comparison (zone, digit, or character) in SELCOD.

- Flags error GP05 if C, Z, or D has not been coded for the type of comparison.

- Moves the indicated character, zone, or digit into SELCHR.

Input:

- Register 2 addresses detail or master record selector information.

Output:

- Binary SELPOS field.
- SELCOD field.
- SELCHR field.
- Erros GP04, GP05, and GP09 flagged in ERTAB1 if errors detected.

Routines Called: None

Print a Record Routine

Entry Point: PRTSYS

Functions:

- Prints the record in the logical record buffer (LRBPRT).
- Ends the job if a printer error occurs.
- Skips to a new page if printer overflow occurs.
- Sets the logical record buffer to blanks.

Input:

- Logical record buffer (LRBPRT).
- SYSPRT parameter list (PRTPRM).

Output:

- Printed record.

Routines Called: SYSPRT

Print Error Messages Routine

Entry Point: ERRPRT

Functions:

- Builds error messages for those flagged in ERTAB1.
- Prints the messages using the Print a Record Routine.
- Indicates if terminal, warning, or informational error messages have been printed.

Input:

- Error table one (ERTAB1)
- Error table two (ERTAB2) addressed by register 2.

Output:

- Printed error messages
- Indication of terminal, warning, and informational error occurrences.

Routines Called: None

Execution Phase (GPEXC)

Entry Point: GPEXC

Chart: DB

Functions:

- Determines the type of gangpunching specified: Intermixed, Match-field, or Count-controlled.

  If intermixed gangpunching is specified:

  - Selects master and detail records from the primary file.
  - Checks that the first record read is a valid master record.
  - After each selected master record, punches and interprets selected detail records that follow it.
  - If the counter is used, updates it after each detail record is punched and resets it after each master record is selected.

  If match-field gangpunching is specified:

  - Selects detail records from the primary file; master records from the secondary file.
  - After each detail or master record is selected, checks for a match-field sequence error.
  - Compares master and detail record match fields.
  - If the match fields are equal, updates the counter (if used) if a detail record has been selected; resets the counter (if used) if a master record has been selected; and punches and interprets the detail record.
  - If match fields are not equal, determines whether a new master or a new detail record should be read.

  If count controlled gangpunching is specified:

  - Selects detail records from the primary file; master records from the secondary file.
  - After a master record is selected, selects detail records, updates the counter, and punches and interprets each detail record selected until counter overflow occurs.

Input:

- FDP table
- Common area of storage

Output:

- Punched and interpreted detail records.

Routines Called: None

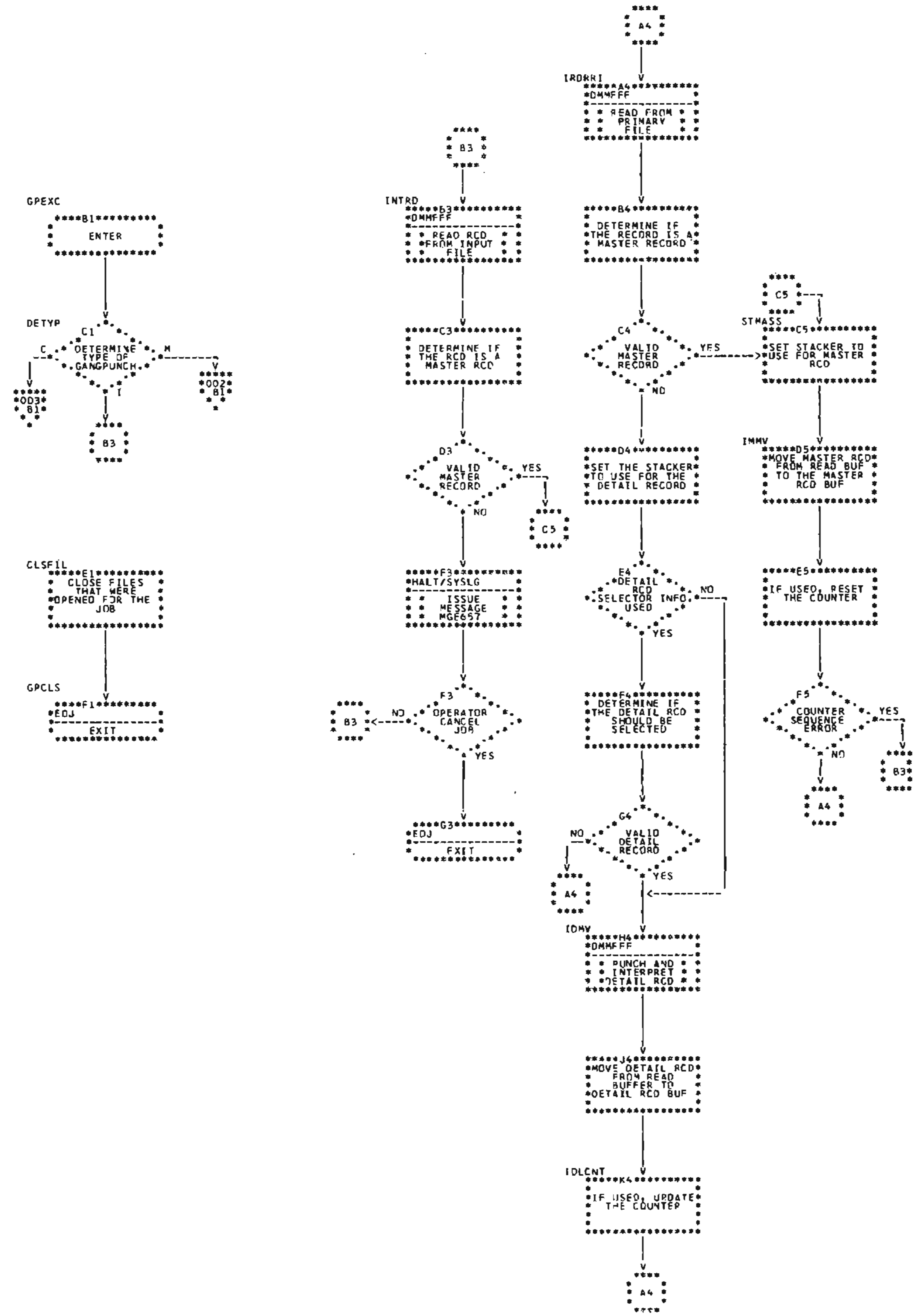
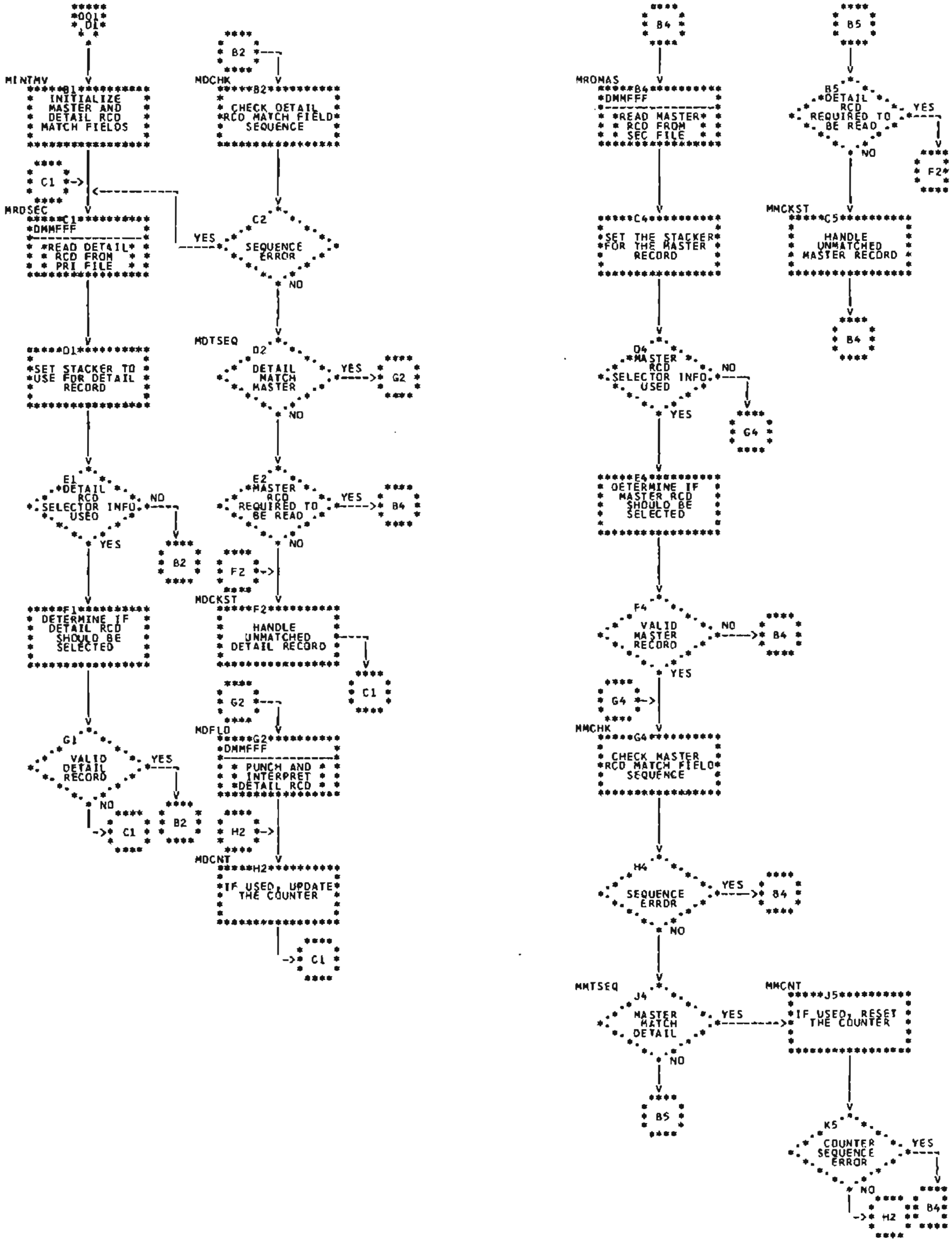

GPEXC
B1
ENTER
DETYP
C1
DETERMINE TYPE OF GANGPUNCH
C
M
I
003 B1
002 B1
B3
CLSFIL
E1
CLOSE FILES THAT WERE OPENED FOR THE JOB
GPCLS
F1
EOJ
EXIT
B3
INTRD
B3
DMMFFF
READ RCD FROM INPUT FILE
C3
DETERMINE IF THE RCD IS A MASTER RCD
D3
VALID MASTER RECORD
YES
NO
C5
E3
HALT/SYSLG
ISSUE MESSAGE MGE657
F3
OPERATOR CANCEL JOB
NO
YES
B3
G3
EOJ
EXIT
A4
IRDRRI
A4
DMMFFF
READ FROM PRIMARY FILE
B4
DETERMINE IF THE RECORD IS A MASTER RECORD
C4
VALID MASTER RECORD
YES
NO
D4
SET THE STACKER TO USE FOR THE DETAIL RECORD
E4
DETAIL RCD SELECTOR INFO. USED
NO
YES
F4
DETERMINE IF THE DETAIL RCD SHOULD BE SELECTED
G4
VALID DETAIL RECORD
NO
YES
A4
IDMV
H4
DMMFFF
PUNCH AND INTERPRET DETAIL RCD
J4
MOVE DETAIL RCD FROM READ BUFFER TO DETAIL RCD BUF
IDLCNT
K4
IF USED, UPDATE THE COUNTER
A4
C5
STMASS
C5
SET STACKER TO USE FOR MASTER RCD
IMMV
D5
MOVE MASTER RCD FROM READ BUF TO THE MASTER RCD BUF
E5
IF USED, RESET THE COUNTER
F5
COUNTER SEQUENCE ERROR
YES
NO
B3
A4

Chart DB. Execution Phase (GPEXC) (Part 1 of 3)



001 D1
B2
MINTMV B1 INITIALIZE MASTER AND DETAIL RCD MATCH FIELDS
MDCHK B2 CHECK DETAIL RCD MATCH FIELD SEQUENCE
C1
MRDSEC C1 DMMFFF READ DETAIL RCD FROM PRI FILE
C2 SEQUENCE ERROR
YES
NO
D1 SET STACKER TO USE FOR DETAIL RECORD
MDTSEQ D2 DETAIL MATCH MASTER
YES G2
NO
E1 DETAIL RCD SELECTOR INFO. USED
NO B2
YES
E2 MASTER RCD REQUIRED TO BE READ
YES B4
NO
F2
MDCKST F2 HANDLE UNMATCHED DETAIL RECORD
C1
F1 DETERMINE IF DETAIL RCD SHOULD BE SELECTED
G2
MDFLD G2 DMMFFF PUNCH AND INTERPRET DETAIL RCD
G1 VALID DETAIL RECORD
YES B2
NO C1
H2
MDCNT H2 IF USED, UPDATE THE COUNTER
C1
B4
MRDMAS B4 DMMFFF READ MASTER RCD FROM SEC FILE
C4 SET THE STACKER FOR THE MASTER RECORD
D4 MASTER RCD SELECTOR INFO. USED
NO G4
YES
E4 DETERMINE IF MASTER RCD SHOULD BE SELECTED
F4 VALID MASTER RECORD
NO B4
YES
G4
MMCHK G4 CHECK MASTER RCD MATCH FIELD SEQUENCE
H4 SEQUENCE ERROR
YES B4
NO
MMTSEQ J4 MASTER MATCH DETAIL
YES
NO
B5
B5
B5 DETAIL RCD REQUIRED TO BE READ
YES F2
NO
MMCKST C5 HANDLE UNMATCHED MASTER RECORD
B4
MMCNT J5 IF USED, RESET THE COUNTER
K5 COUNTER SEQUENCE ERROR
YES B4
NO H2

Chart DB. Execution Phase (GPEXC) (Part 2 of 3)

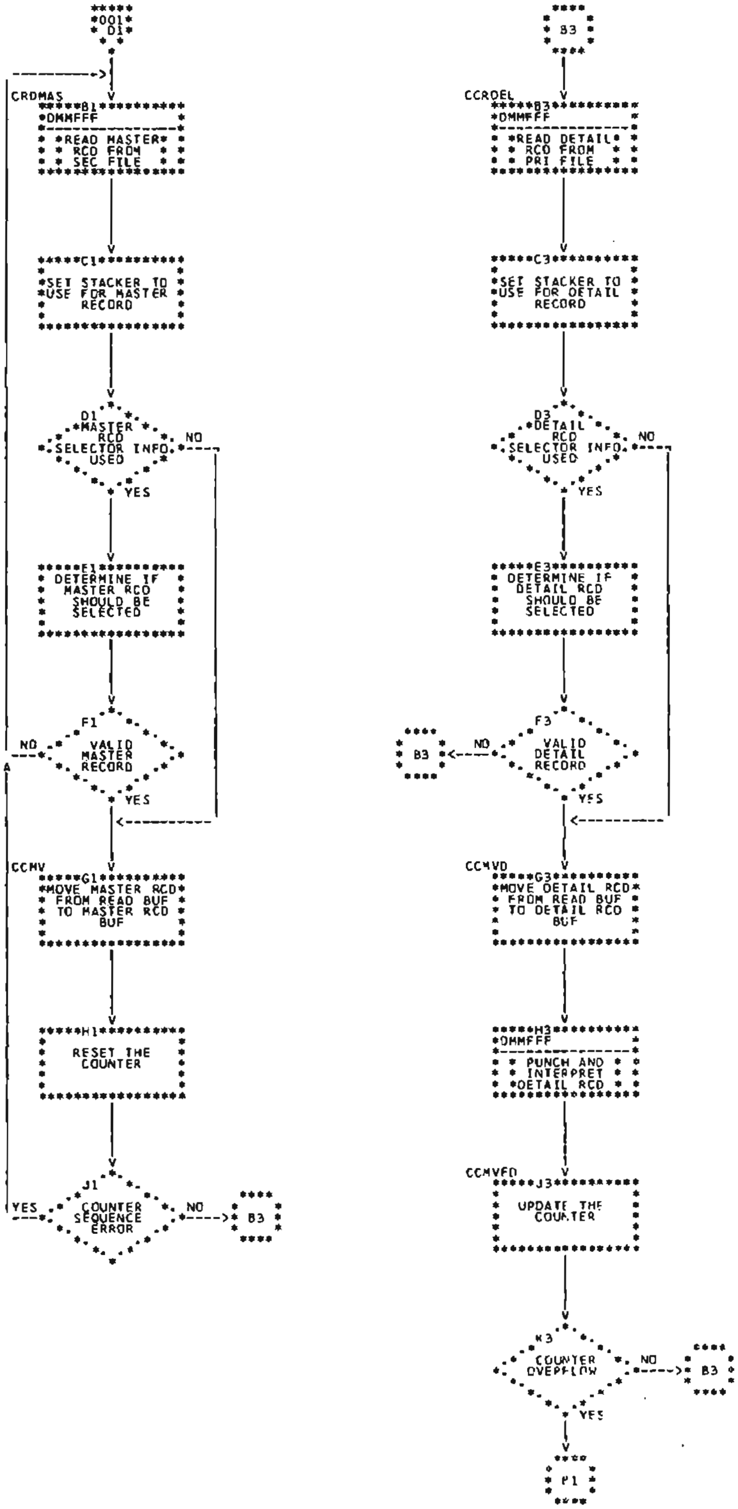

001
D1
CRDMAS
B1
DMMFFF
READ MASTER RCD FROM SEC FILE
C1
SET STACKER TO USE FOR MASTER RECORD
D1
MASTER RCD SELECTOR INFO. USED
NO
YES
E1
DETERMINE IF MASTER RCD SHOULD BE SELECTED
F1
VALID MASTER RECORD
NO
YES
CCMV
G1
MOVE MASTER RCD FROM READ BUF TO MASTER RCD BUF
H1
RESET THE COUNTER
J1
COUNTER SEQUENCE ERROR
YES
NO
B3
B3
CCRDEL
B3
DMMFFF
READ DETAIL RCD FROM PRI FILE
C3
SET STACKER TO USE FOR DETAIL RECORD
D3
DETAIL RCD SELECTOR INFO. USED
NO
YES
E3
DETERMINE IF DETAIL RCD SHOULD BE SELECTED
F3
VALID DETAIL RECORD
B3
NO
YES
CCMVD
G3
MOVE DETAIL RCD FROM READ BUF TO DETAIL RCD BUF
H3
DMMFFF
PUNCH AND INTERPRET DETAIL RCD
CCMVFD
J3
UPDATE THE COUNTER
K3
COUNTER OVERFLOW
NO
B3
YES
P1

Chart DB. Execution Phase (GPEXC) (Part 3 of 3)

Update the Counter Routine

Entry Point: UPDAT

Functions:

- Adds one to the counter if ascending sequence is specified.
- Subtracts one from the counter if descending sequence is specified.
- Indicates counter overflow.

Input:

- Current counter value
- Counter sequence

Output:

- Updated counter
- Counter overflow indication

Routines Called: None

Reset the Counter Routine

Entry Point: RESET

Functions:

- Resets the counter to its starting value (either the constant value specified in CNTSTR or a variable value specified on the master record).
- If a variable starting and/or ending counter value is specified, checks the counter limits for a sequence error.

Input:

- Constant counter starting value or master record positions in which the counter starting value is found.
- Constant counter ending value or master record positions in which the counter ending value is found.

Output:

- Reset counter starting value
- Reset counter ending value if a variable end value has been specified

Routines Called: None

Sequence Check the Match Field Routine

Entry Point: SEQCK

Functions:

- For detail records, compares the match field of the record just read (in the read buffer) with the match field of the previous record read (in the detail record buffer). Logs a DETAIL FILE SEQUENCE ERROR message and indicates the error condition in SWTCH2 if the fields are out of the expected sequence.
- For master records, compares the match field of the record just read (in the read buffer) with the match field of the previous record read (in the master record buffer). Logs a MASTER FILE SEQUENCE ERROR message and indicates the error condition in SWTCH2 if the fields are out of the expected sequence.

Input:

- Read buffer contents.

- Detail or master record buffer contents.

Output:

- If a sequence error is detected, error message and error indication in SWTCH2. Otherwise, none.

Routines Called: None

Determine Record Type Routine

Entry Point: DRTSUB

Function:

- Determines if the record just read (in the read buffer) is valid according to the selector information specified in either columns 9 to 18 or 19 to 28 of the header record.

Input:

- Read buffer contents

- MASSL1 and MASSL2 or DETSL1 and DETSL2 selector information from the common area of storage.

Output:

- Result of test indicated in SWTCH2.

Routines Called: None

Build the Output Record Routine

Entry Point: MVFLD

Functions:

- If the entire detail record is to be printed, moves the detail record to the print buffer.

- Processes the entire FDP table, one entry at a time to build the output record for each detail record.

- For each FDP table entry, determines the entry type and then:
  - For an M-type entry, moves the specified master record information into the detail record punch and print buffers.
  - For an X-type entry, moves the counter to the detail record punch and print buffers.
  - For a C-type entry, moves the specified constant to the detail record punch and print buffers.

Input:

- Detail record buffer

- FDP table

- Common area of storage

Output:

- Detail record punch and print buffers

Routines Called: None

I/O Interface Routine

Entry Point: IOSUB

Functions:

- Accesses the proper data management routine to perform the requested I/O function (open, close, read, punch, or print).
- Indicates end-of-job.

Input:

- XR2 contains the operation to be performed

Output:

- One of the following:
  - Opened file
  - Closed file
  - Record in read buffer
  - Punched detail record
  - Interpreted detail record
  - End-of-file indication

Routines Called: Proper data management routine.

Section 4. Data Area Formats

This section describes data areas that are used by two or more routines.

Common Area

The common area is a 56-byte area following the FDP table that indicates the following:

- Buffer addresses
- FDP table address
- Header record information
- Gangpunching errors
- Valid records to be selected
- Records to be checked for sequence errors
- Current counter value
- File operation codes
- I/O device type

The common area is loaded with the diagnostic phase into storage. Figure 4-2 shows the format and contents of this area.

Displacement of leftmost byte in hexadecimal	Label	Length in bytes	Description	Routines that change data
0	MASBUF	2	Address of the master record buffer	–
2	DETBUF	2	Address of the detail record buffer	–
4	PUNBUF	2	Address of the punch buffer	–
6	PRTBUF	2	Address of the print buffer	–
8	READBF	2	Address of the read buffer	IOSUB (GPEXC)
A	FLDDF	2	Address of the FDP table	–
C	MASTSS	1	Stacker to use for the master record	GANGP
D	DETSS	1	Stacker to use for the detail record	GANGP
E	MASTNM	1	Stacker to use for a master record that does not match a detail record	GANGP
F	DETNM	1	Stacker to use for a detail record that does not match a master record	GANGP

10	GPTYPE	1	Type of gangpunching:
			I = Interspersed
			M = Match-field
			C = Count-controlled
				GANGP

11	MATCHL	1	Length minus one of the match field	GANGP
12	MATCHM	1	End position minus one of the match field in the master record	GANGP
13	MATCHD	1	End position minus one of the match field in the detail record	GANGP
14	SWTCH1	1	Flag byte X'80' = On – digit comparison Off – character comparison X'40' = On – match fields in descending sequence Off – match fields in ascending sequence X'20' = Stop on an unmatched detail or master record	GANGP

Figure 4-2 (Part 1 of 6). Common Area

Displacement of leftmost byte in hexadecimal	Label	Length in bytes	Description	Routines that change data
			X'10' = The counter is used	
			X'08' = On — the counter descending Off — the counter ascending	
			X'04' = Variable counter starting value	
			X'02' = Variable counter ending value	
			X'01' = Print the option specified	
15	SWTCH2	1	Flagbyte	
			X'80' = Print only what is punched into the detail record	GANGP
			X'40' = Counter overflow	UPDAT (GPEXC)
			X'20' = Counter sequence error when variable counter limits specified	RESET (GPEXC)
			X'10' = Match field sequence error	SEQCK (GPEXC)
			X'03' = Valid record which should be selected	DRTSUB (GPEXC)
			X'04' = On — Check the detail record match field for a sequence error Off — Check the master record match field for a sequence error	GPEXC
			X'02' = Match indicator	MATPRL (GPEXC)
			X'01' = End of file indicator	IOSUB (GPEXC)
16	CNTSTR	4	Counter starting value (constant)	GANGP RESET (GPEXC)
1A	CNTEND	4	Counter ending value (constant)	GANGP RESET (GPEXC)
1E	COUNTR	4	Current counter value	UPDAT (GPEXC) RESET (GPEXC)
22	CNTLNG	1	Counter length minus one	

Figure 4-2 (Part 2 of 6). Common Area

Displacement of leftmost byte in hexadecimal	Label	Length in bytes	Description	Routines that change data
23	MASSL1	3	Master record selector information *Byte* *Meaning* 0 End position minus one in the master record of the character to be compared 1 Flag byte X'80' = Character comparison X'40' = Zone comparison X'20' = Digit comparison On — test for equal comparison Off — test for unequal comparison Bits 4-7 Not used 2 Character to be compared with the master record character If master record selector information is not specified, byte 0 of MASSL1 is set to X'FF'	GANGP
26	MASSL2	3	Master record selector information *Byte* *Meaning* 0 End position minus one in the master record of the second character to be compared. 1 Flag byte X'80' = Character comparison X'40' = Zone comparison X'20' = Digit comparison On — test for equal comparison Off — test for unequal comparison Bits 4-7 Not used 2 Second character to be compared with the master record character	GANGP

Figure 4-2 (Part 3 of 6). Common Area

Displacement of leftmost byte in hexadecimal	Label	Length in bytes	Description	Routines that change data
			If MASSL2 master record selector information is not specified, byte 0 of MASSL2 is set to X'FF'	
27		1	X'FF'. Indicates the end of master record selector information if both MASSL1 and MASSL2 are specified.	
2A	DETSL1	3	Detail record selector information	GANGP
			Byte *Meaning*	
			0 End position minus one in the detail record of the character to be compared	
			1 Flag byte X'80' = Character comparison X'40' = Zone comparison X'20' = Digit comparison On – test for equal comparison Off – test for unequal comparison Bits 4-7 Not used	
			2 Character to be compared with the detail record character	
			If detail record selector information is not specified, byte 0 of DETSL1 is set to X'FF'	
2D	DETSL2	3	Detail record selector information	GANGP
			Byte *Meaning*	
			0 End position minus one in the detail record of the second character to be compared.	
			1 Flag byte	

Figure 4-2 (Part 4 of 6). Common Area

Displacement of leftmost byte in hex-adecimal	Label	Length in bytes	Description	Routines that change data
			X'80' = Character comparison	
			X'40' = Zone comparison	
			X'20' = Digit comparison On — test for equal comparison Off — test for unequal comparison	
			Bits 4-7 Not used	
			2 Second character to be compared with the detail record character	
			If DETSL2 detail record selector information is not specified, byte 0 of DETSL2 is set to X'FF'	
30		1	X'FF'. Indicates the end of detail record selector information if both DETSL1 and DETSL2 are specified.	
31	SECFOP	1	Flag byte for the secondary file:	IOSUB (GPEXC)
			0 X'80' = Off indicating a secondary file operation	
			1 X'40' = Read	
			4 X'08' = First end-of-file indicator	
			5 X'04' = Open	
			6 X'08' = Close	
32	SECSTK	1	Indicates the stacker to be used for the secondary file.	GANGP GPEXC
33	PRIFOP	1	Flag byte for the primary file:	IOSUB (GPEXC)
			X'80' = On indicating a primary file operation	
			X'40' = Read	
			X'80' = Punch	
			X'10' = Print	MVFLD
			X'08' = First end-of-file indicator	
			X'04' = Open	
			X'08' = Close	

Figure 4-2 (Part 5 of 6). Common Area

Displacement of leftmost byte in hexadecimal	Label	Length in bytes	Description	Routines that change data
34	PRISTK	1	Indicates the stacker to be used for the primary file	GANGP GPEXC
35	CNTST	1	End position minus one of the counter starting value (variable) in the master record	GANGP
36	CNTEN	1	End position minus one of the counter ending value (variable) in the master record	GANGP
37	DEVTYP	1	Device type X'80' = 5424 is used	

Figure 4-2 (Part 6 of 6). Common Area

Define the File – GPDTF1 (diagnostic phase), PRIDTF and SECDTF (execution phase).

These are 37-byte parameter lists which are passed to the I/O device IOS routine. They contain operation codes to indicate the device functions to be performed. Each has the format shown in Figure 4-3.

NAME	OFFSET	LENGTH	CONTENTS
MDFDEV	0	1	Device address (first 5 bits of Q code) X'F0' = Primary hopper X'F8' = Secondary hopper
MDFUPS	1	1	External indicator
MDFAT1	2	1	Attribute byte 1 Bit 0 = Input Bit 1 = Output Bit 4 = Print
MDFAT2	3	1	Attribute byte 2 Bit 0 = End of file on last read Bit 1 = File allocated Bit 3 = Dual I/O areas Bit 5 = Hopper used as system input device Bit 6 = /& read on last input operation Bit 7 = File is opened
MDFCHA	5	2	DTF chain pointer A
MDFCHB	7	2	DTF chain pointer B
MDFARR	9	2	ARR save area (return address)
MDFXR1	B	2	XR1 save area (contents of object program XR1)
MDFLRA	D	2	Logical record address
MDFCMP	E	1	Completion code X'40' = Normal completion X'41' = Abnormal condition X'42' = End of file indicator
MDFOPR	F	1	Operation Bit 0 = Read Bit 1 = Print Bit 2 = Punch Bit 3 = Move (deferred operation)
MDFSTS	10	1	Stacker select Bit 2 = Print 4 lines Bit 5 = Select stacker Stacker 1 2 3 4 Bit 6 = 0 1 1 0 Bit 7 = 1 0 1 0
MDFQ	11	1	Q byte (device address)
MDFR	12	1	R byte
	13	1	Not used
MDFWKA	16	3	Work area
MDFSVA	18	2	Address of 15-byte permanent save area
MDFERP	1A	2	Pointer to ERP
MDFRIO	1C	2	Address of current read IOB
MDFUIO	1E	2	Address of current punch IOB (not referenced)
MDFPUB	20	2	Address of current punch I/O area
MDFPTB	22	2	Address of print IOB
MDFPTL	23	1	Print buffer length (not referenced)
MDFPUL	24	1	Punch buffer length (not referenced)

Figure 4-3. Define the File

Error Table 1 — ERTAB1

ERTAB1 is a 76-byte table used by the diagnostic phase that indicates the following:

- The errors that have occurred
- The type of error that has occurred (terminal, warning, informational, or action required)
- The errors that should be printed
- The end address of the message to be printed (in ERTAB2) for the error

Figure 4-4 shows the format of this area.

Displacement of leftmost byte in hexadecimal	Label	Length in bytes	Description	Routines that change data
0	ERTAB1	76	ERTAB1 consists of 25 3-byte entries. Each entry has the following format: *Byte* *Contents* 0 Flag byte X'80' = Terminal error X'40' = Warning error X'20' = Informational message X'10' = Action required X'01' = Print the message 1-2 Address of the end of the message text in ERTAB2. The end of ERTAB1 is indicated by X'FF' following the last entry.	GANGP CFCONT IDCHK

Figure 4-4. Error Table 1

FDP Table — FDPADS

The FDP table is a 512 byte storage area between the print buffer and the common area of storage that is built by the diagnostic phase. One FDP table entry is created for each field definition record read. The entries are variable length, depending on the type of field definition record read. Figure 4-5 shows the possible formats of the entries. The end of the FDP table is indicated by X'FF' following the last entry.

The FDP table is used by the execution phase to build the output record for each selected detail record. Once a detail record is selected, the output record is built by processing the entire FDP table first entry to last. The detail record is then punched and interpreted.

Field Definition Record Type	**FDP Entry Format**	
M	*Byte*	*Contents*
	0	Character M
	1	End position minus one of the field in the detail record in which master record information will be punched
	2	Length minus one of the field to be punched in the detail record
	3	End position minus one of the field in the master record that is to be punched in the detail record
X	*Byte*	*Contents*
	0	Character X
	1	End position minus one of the field in the detail record in which the counter value will be punched
C	*Byte*	*Contents*
	0	Character C
	1	End position minus one of the field in the detail record in which the constant will be punched.
	2	Length minus one of the constant to be punched
	3-n	Constant (from 4 to 62 bytes) that will be punched in the detail record

Figure 4-5. FDP Table Entry Formats

Halt/Syslog Message Tables

HALST1

This 7-byte area is passed to the Halt/Syslog routine and will appear on the Message Display Unit if warning errors have occurred during the diagnostic phase. It indicates the halt to be displayed, the address of the halt message (REVIEW WARNING MESSAGES), and the options that may be selected. When this message is issued, the operator can either cancel the job or continue with the execution of the Gangpunch program.

HALST2

This 7-byte area is passed to the Halt/Syslog routine and will appear on the Message Display Unit if terminal errors have occurred during the diagnostic phase. It indicates the halt to be displayed, the address of the halt message (ERRORS IN SPECIFICATIONS), and the option that may be selected. When this message is issued, the operator must select the option to cancel the program.

MGE650

This 7-byte area is passed to the Halt/Syslog routine and will appear on the Message Display Unit if a master record match field is out of the expected sequence. It indicates the halt to be displayed, the address of the halt message (MASTER FILE SEQUENCE ERROR), and the options that may be selected. When this message is issued, the operator can either cancel the job or continue with the execution of the Gangpunch program.

MGE651

This 7-byte area is passed to the Halt/Syslog routine and will appear on the Message Display Unit if a detail record match field is out of the expected sequence. It indicates the halt to be displayed, the address of the halt message (DETAIL FILE SEQUENCE ERROR), and the options that may be selected. When this message is issued, the operator can either cancel the job or continue with the execution of the Gangpunch program.

MGE652

This 7-byte area is passed to the Halt/Syslog routine and will appear on the Message Display Unit if an unmatched master record is found during match-field gangpunching. It indicates the halt to be displayed, the address of the halt message (UNMATCHED MASTER CARD), and the option that may be selected. When this message is issued, the operator must select the option to continue with the execution of the Gangpunch program.

MGE653

This 7-byte area is passed to the Halt/Syslog routine and will appear on the Message Display Unit if an unmatched detail record is found during match-field gangpunching. It indicates the halt to be displayed, the address of the halt message (UNMATCHED DETAIL RECORD), and the option that may be selected. When this message is issued, the operator must select the option to continue with the execution of the Gangpunch program.

MGE654

This 7-byte area is passed to the Halt/Syslog routine and will appear on the Message Display Unit if the variable counter start and/or end limits are incorrect for the sequence specified. It indicates the halt to be displayed, the address of the halt message (INVALID VARIABLE COUNTER), and the options that may be selected. When this message is issued, the operator can either cancel the job or continue with the execution of the Gangpunch program.

MGE657

This 7-byte area is passed to the Halt/Syslog routine and will appear on the Message Display Unit if a master record is expected to be read but not found. It indicates the halt to be displayed, the address of the halt message (MASTER CARD MISSING), and the options that may be selected. When this message is issued, the operator can either cancel the job or continue with the execution of the Gangpunch program.

Logical Record Buffer — LRBPRT

This is a 132-byte area that is used as an output buffer by the SYSPRT routine to print the following:

- Gangpunching heading line
- Header record
- S error indicators beneath the header record
- Field definition record
- Errors diagnosed during header and field definition record processing

This buffer is used only during the diagnostic phase.

Physical Print Buffer — PPNTBF

This 256-byte area contains the information to be printed on the detail record. The buffer is created by the Build an Output Record routine (MVFLD), and the detail record is printed by the proper data management routine in the I/O Interface routine (IOSUB). The address of this area is contained in PRTBUF.

Physical Punch Buffer — PPCHBF

This 128-byte area contains information to be punched in the detail record. The buffer is created by the Build an Output Record routine (MVFLD), and the detail record is punched by the proper data management routine in the I/O Interface routine (IOSUB). The address of this area is contained in PUNBUF.

Physical Read Buffer — PREDBF

This 96-byte area contains master and detail records read during the gangpunch execution phase. The data management routine for the device is given control (in the IOSUB routine) to read the master or detail record into this buffer. The address of this area is stored in READBF.

Read Buffer — GPRD1

This 96-byte area contains the header and field definition records read from the MFCU. This buffer is used only during the diagnostic phase.

Part V. Data Recording

Section 1. Introduction

The IBM System/3 Data Recording program causes the IBM System/3 to function as if it were the IBM 5496 Data Recorder. In simulating the Data Recorder, this program accepts control cards that specify the format of the card image before the card is punched and printed.

System Requirements

The IBM System/3 Data Recording program operates using the following system configurations:

- The IBM 5410 Processing Unit.
- The IBM 5424 Multi-Function Card Unit (MFCU).
- The IBM 5475 Data Entry Keyboard.
- The IBM 5444 Disk Storage Drive.

Section 2. Method of Operation

This section is concerned with the functional flow of logic and data for the IBM System/3 Data Recording program. The following section, *Section 3, Program Organization,* will expand upon the items found in this functional overview.

General Flow of the Data Recording Program

After the Data Recording program is loaded, the Initializing routine senses the 5475 Data Entry Keyboard status into the sense table (SNS instruction). Data areas are then cleared or initialized to a predetermined setting (column indicator set to 01). The program then waits for an interrupt from the Data Entry Keyboard.

When an interrupt is detected (see Figure 5-1), control is passed to the Interrupt Handler routine which disables any further interrupts until the current one is resolved. The Interrupt Handler routine then senses the current status of the Data Entry Keyboard into the sense table. Control is then passed to the Interrupt Service routine which tests the Sense Table, determines the source of the interrupt, and passes control to the respective routine to service the interrupt.

Depending upon the type of interrupt, four general types of actions can result:

- Type A: The assembly area is placed under program control.
- Type B: The format of the card image (in the assembly area) is modified in accordance with the program control card entries.
- Type C: Data is entered into the assembly area.
- Type D: The card to be punched is released, and the information contained in the assembly area is punched and printed on that card.

After an interrupt is serviced, control is passed back to the Return routine, which waits for the next interrupt.

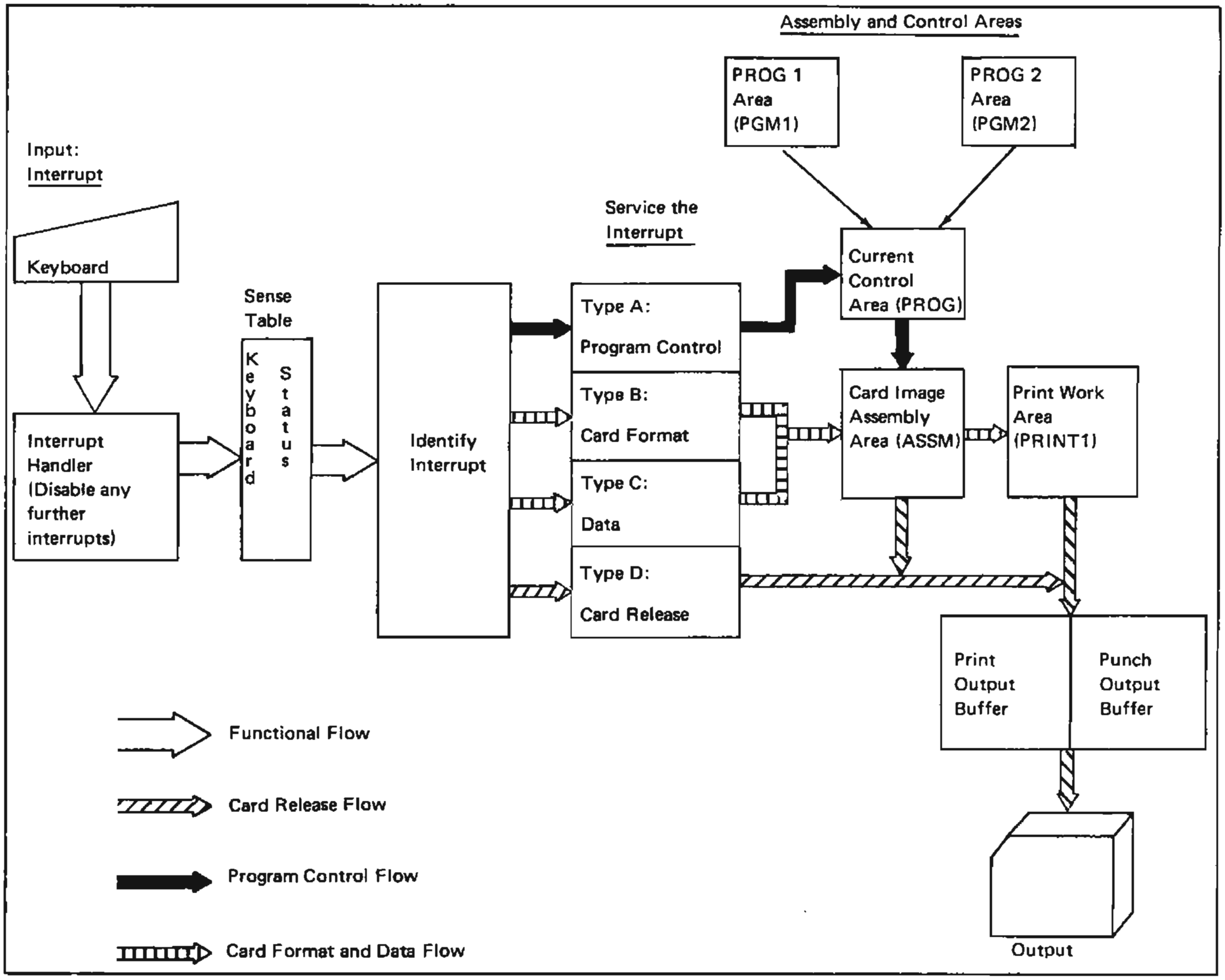


Figure 5-1. Functional Flow of Data and Control for Data Recording Program

Section 3. Program Organization

This section is designed to show how the routines that comprise the Data Recording program are interconnected.

Figure 5-2 shows the general layout of the separate routines. The text that follows the figure explains the function of each routine. Flowcharts are included for routines where complexity warrants flowcharting. Figure 5-3 contains a storage map for the Data Recording program.

Initializing Routine

Entry Point: ADRAA1

Chart: None

Function:

- Calls Program Protect transient routine which checks for copyright violation.
- Loads register 1 with base address.
- Loads the level 1 instruction address register (IAR) with address of Interrupt Handler routine.
- Loads register 2 with base address.
- Senses the keyboard status into the sense table.
- Reads the first card to be punched from the secondary hopper.

Display Routine

Entry Point: AAB010

Chart: None

Function:

- Loads the LITE data area for the Return routine by indexing TAB data area with the decimal value of DCNT.

Return Routine

Entry Point: AAC000

Chart: None

Function:

- Displays column indicator (LITE) data area.
- Displays on-off status of program 1 and/or program 2.
- Completes current interrupt and enables further interrupts by giving a SNS instruction.
- When an interrupt occurs, control is passed to the interrupt handler via a hardward exchange of IAR's.
- Passes control to the Interrupt Service routine (AAE010) after returning from the Interrupt Handler routine.

Interrupt Handler Routine

Entry Point: AAD010

Chart: None

Function: When an interrupt occurs, there is a physical exchange in the level 1 IAR instruction address register passing control to the Interrupt Handler routine. The Interrupt Handler routine performs the following:

- Disables any further interrupts by use of an SIO instruction.
- Senses the keyboard status into the sense table.
- Returns control to the Return routine.

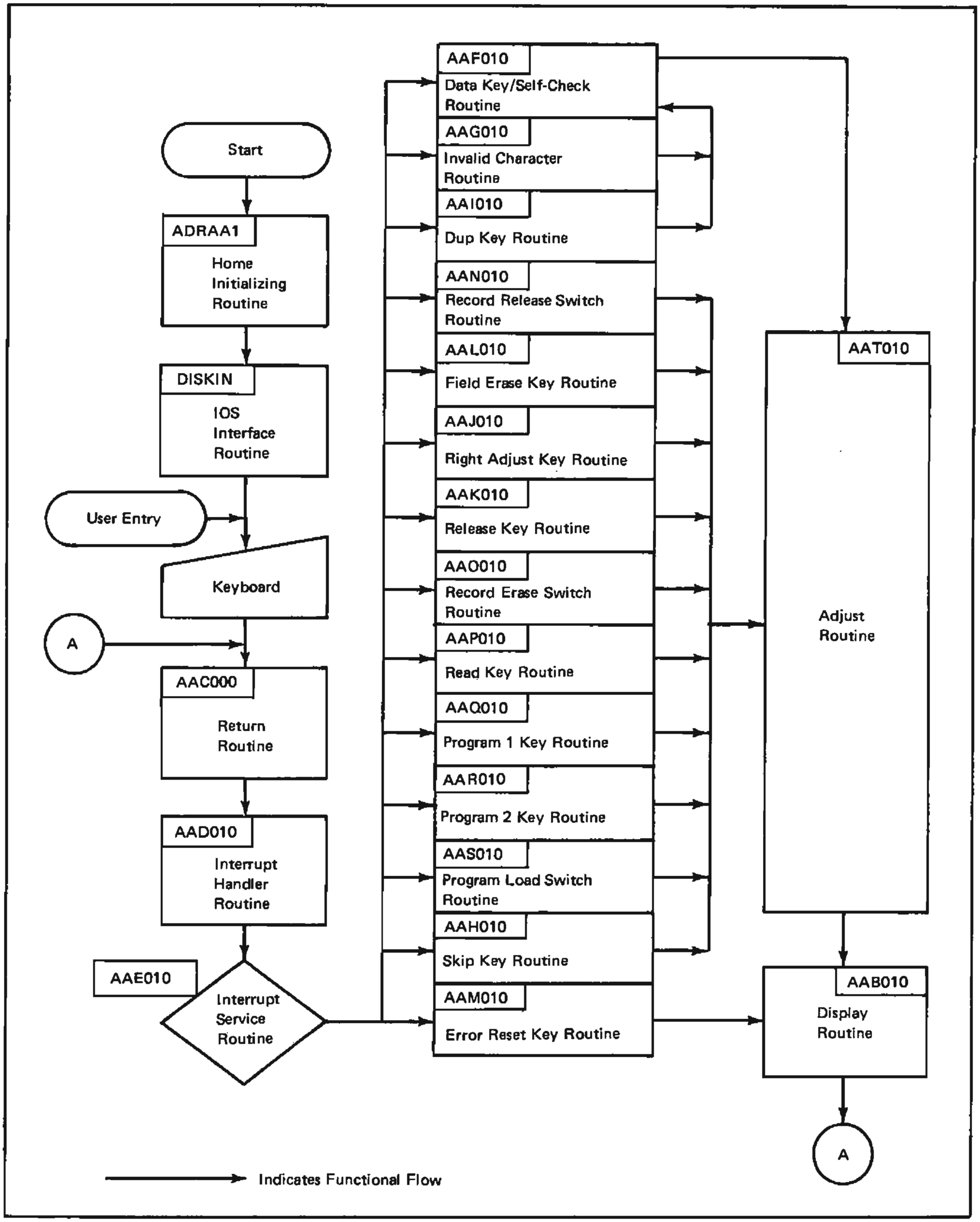


Figure 5-2. Program Organization of Data Recording Program

Supervisor (System Communication Region)
RDBUF (Read Buffer)
PRINT0 (Print Work Area)
Copyright
Work Areas and Status Table
HOLD (Hold Area)
ASSM (Assembly Area)
Stick Table, Counters, Program Area
PRINT1 (Print Buffer)
Program Control Areas
Initializing Routine
Column Indicator Display
Return Routine
Interrupt Handler Routine
Interrupt Service Routine
Data Key/Self Check Routine
Invalid Character Routine
Skip Key Routine
DUP Key Routine
Right Adjust Key Routine
Release Key Routine
Field Erase Key Routine
Error Reset Key Routine
Record Release Switch Routine
Record Erase Switch Routine
Read Key Routine
PROG1 Key Routine
PROG2 Key Routine
Program Load Switch Routine
Adjust Routine
End or Beginning of Field Routine
Test for Auto Skip Field Routine
IOS Interface Routine
Full Function MFCU IOS Routine

Figure 5-3. Storage Map for Data Recording Program

Interrupt Service Routine

Entry Point: AAE010

Chart: None

Function: Upon receiving control from the Return routine, this routine:

- Tests sense table to locate the source of the interrupt.
- Passes control to the appropriate routine (see Figure 5-2).
- If a source of interrupt is not found or if an invalid interrupt is detected a halt is initiated.

Data Key/Self-Check Routine

Entry Point: AAF010

Chart: GA (parts 1 and 2)

Function:

- Tests for override conditions.
- Moves keyed characters into the assembly area.
- Moves keyed characters into print area if PRINT switch is on.
- Performs self-check function if in self-check field (use self-check modulus 11).

Invalid Character Routine

Entry Point: AAG010

Chart: None

Function: If an invalid character is entered from the Data Entry Keyboard, this routine:

- Locks the Data Entry Keyboard.
- Turns on error light.

Note: An invalid character occurs when the current column was programmed for a numeric shift and a character other than 0-9 or a blank was keyed.

Skip Key Routine

Entry Point: AAH010

Chart: GB

Function:

- Inserts blanks into assembly area (ASSM) on a field basis.
- Inhibits self-check print code 'SC' when skipping a field with a self-check error.

Dup Key Routine

Entry Point: AAI010

Chart: GC

Function:

- Moves the previous card image from the hold area into the assembly area on a column by column basis.
- Tests for override condition.

Right Adjust Key Routine

Entry Point: AAJ010

Chart: GD

Function:

- Shifts keyed data of the current field into the rightmost bytes of that field.
- Fills the vacated, leftmost bytes with blanks.

Release Key Routine

Entry Point: AAK010

Chart: GE

Function:

- Moves the column indicator up through column 96 to column 00 under the following conditions:
  1. Under manual control, the column indicator is moved to 00.
  2. Under program control, if a field is programmed for automatic duplication and the AUTO SK/DUP switch is on, data from the hold area is moved to the assembly area on a column-by-column basis. Otherwise, the column indicator is moved to 00.
- Punches and prints a card.
- Reads next card to be punched.

Field Erase Key Routine

Entry Point: AAL010

Chart: GF

Function: This routine causes the column indicator to backspace to:

- The beginning of the last manual field.
- The beginning of the last keyed word.

Error Reset Key Routine

Entry Point: AAM010

Chart: None

Function:

- Restores operational functions of the Data Entry Keyboard.
- Turns off the error light.

Record Release Switch Routine

Entry Point: AAN010

Chart: None

Function: Sets an internal switch to reflect the current status of the Record Release switch.

Record Erase Switch Routine

Entry Point: AAO010

Chart: None

Function:

- This routine clears the print and assembly areas in storage.
- Restores Data Entry Keyboard to operational status.
- Resets all internal self-check switches.

Read Key Routine

Entry Point: AAP010

Chart: None

Function: Reads a card into the hold area from the primary hopper.

Program 1 Key Routine

Entry Point: AAQ010

Chart: GG

Function:

- Moves the data from the program 1 area (PGM1) to the current control area (PROG).
- Turns on the program 1 bit in the LITE data area.

Program 2 Key Routine

Entry Point: AAR010

Chart: GH

Function:

- Moves the data from the program 2 area (PGM2) to the current control area (PROG).
- Turns on the program 2 bit in the LITE data area.

Program Load Switch Routine

Entry Point: AAS010

Chart: GI

Function:

- Reads a card into the assembly area.
- Checks for an end-of-job card (EOJ).
- Moves card images other than EOJ cards from the assembly area into the current control area (PROG).
- Calls EOJ transient routine when encountering an EOJ card.

Adjust Routine

Entry Point: AAT010

Chart: GJ

Function:

- Updates counters (BCNT and DCNT) to reflect the next column to be worked on.
- Checks for automatic functions and branches to the appropriate routine.

End or Beginning of Program Defined Field Routine

Entry Point:

- AAU010 -- End of field check
- AAU020 -- Beginning of field check

Chart: None

Function:

- Checks for end of field by determining if one of the following is satisfied:
  1. Current column of current control area (PROG) is at end of the record.
  2. Current column of current control area (PROG) contains an end-of-field code.
- Checks for beginning of field by determining if one of the following is satisfied:
  1. Current column of current control area (PROG) is at beginning of the record.
  2. Preceding column of current control area (PROG) contains an end-of-field code.

Test for Auto Skip Field Routine

Entry Point: AAV001

Chart: None

Function: Checks the current column of the current control area (PROG) for any one of the codes which indicates an auto skip field.

IOS Interface Routine

Entry Point: DISKIN

Chart: GK

Function: Initiates MFCU IOS operations by translating an IOCS parameter list and passing it to the Full Function MFCU IOS routine.

Exits:

- Normal – To the routine in main storage that called it via the Address Recall Register.
- Error – To Halt/Syslog.

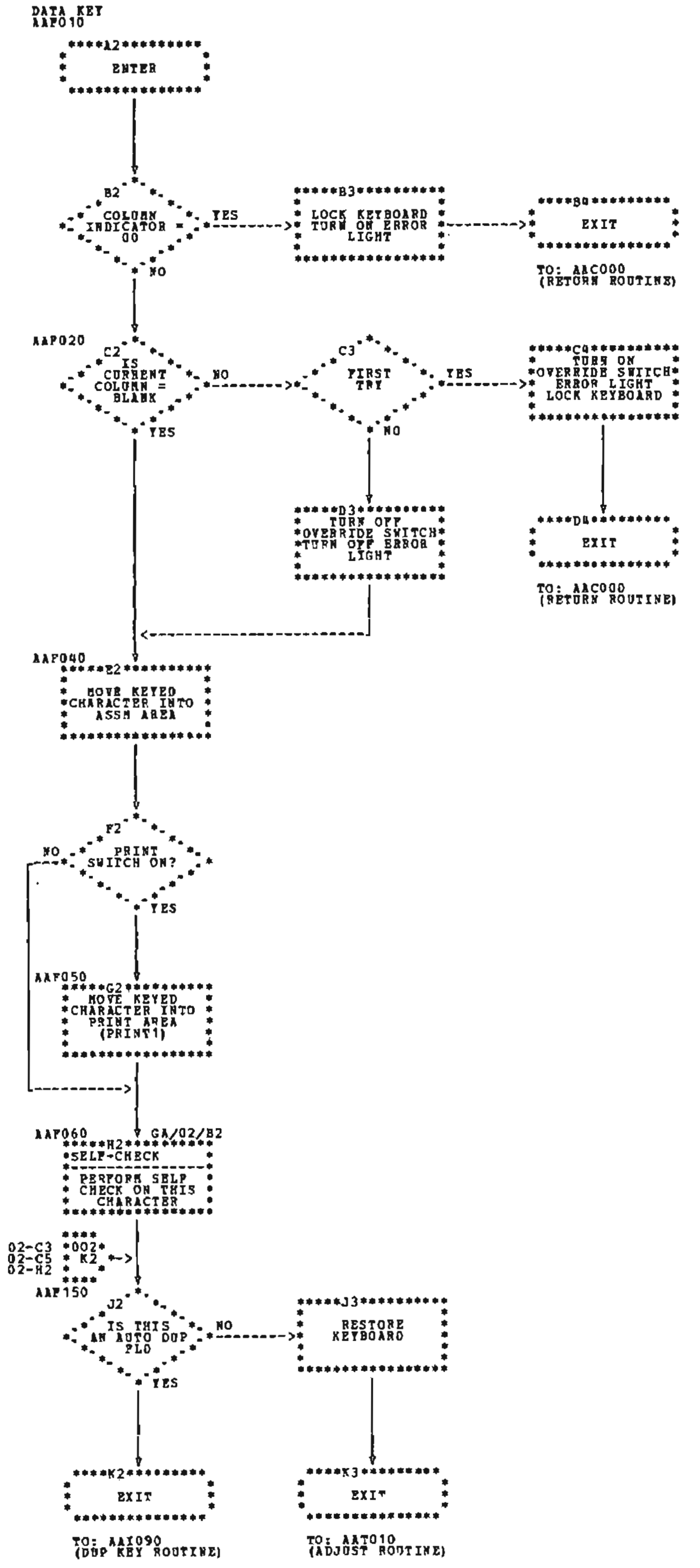

DATA KEY
AAF010
A2
ENTER
B2
COLUMN INDICATOR = 00
YES
NO
B3
LOCK KEYBOARD TURN ON ERROR LIGHT
B4
EXIT
TO: AAC000 (RETURN ROUTINE)
AAF020
C2
IS CURRENT COLUMN = BLANK
NO
YES
C3
FIRST TRY
YES
NO
C4
TURN ON OVERRIDE SWITCH ERROR LIGHT LOCK KEYBOARD
D3
TURN OFF OVERRIDE SWITCH TURN OFF ERROR LIGHT
D4
EXIT
TO: AAC000 (RETURN ROUTINE)
AAF040
E2
MOVE KEYED CHARACTER INTO ASSM AREA
F2
PRINT SWITCH ON?
NO
YES
AAF050
G2
MOVE KEYED CHARACTER INTO PRINT AREA (PRINT1)
AAF060
GA/02/B2
H2
SELF-CHECK
PERFORM SELF CHECK ON THIS CHARACTER
02-C3
02-C5
02-H2
002
K2
AAF150
J2
IS THIS AN AUTO DUP FLD
NO
YES
J3
RESTORE KEYBOARD
K2
EXIT
TO: AAI090 (DUP KEY ROUTINE)
K3
EXIT
TO: AAT010 (ADJUST ROUTINE)

Chart GA. Data Key/Self-Check Routine (Part 1 of 2)

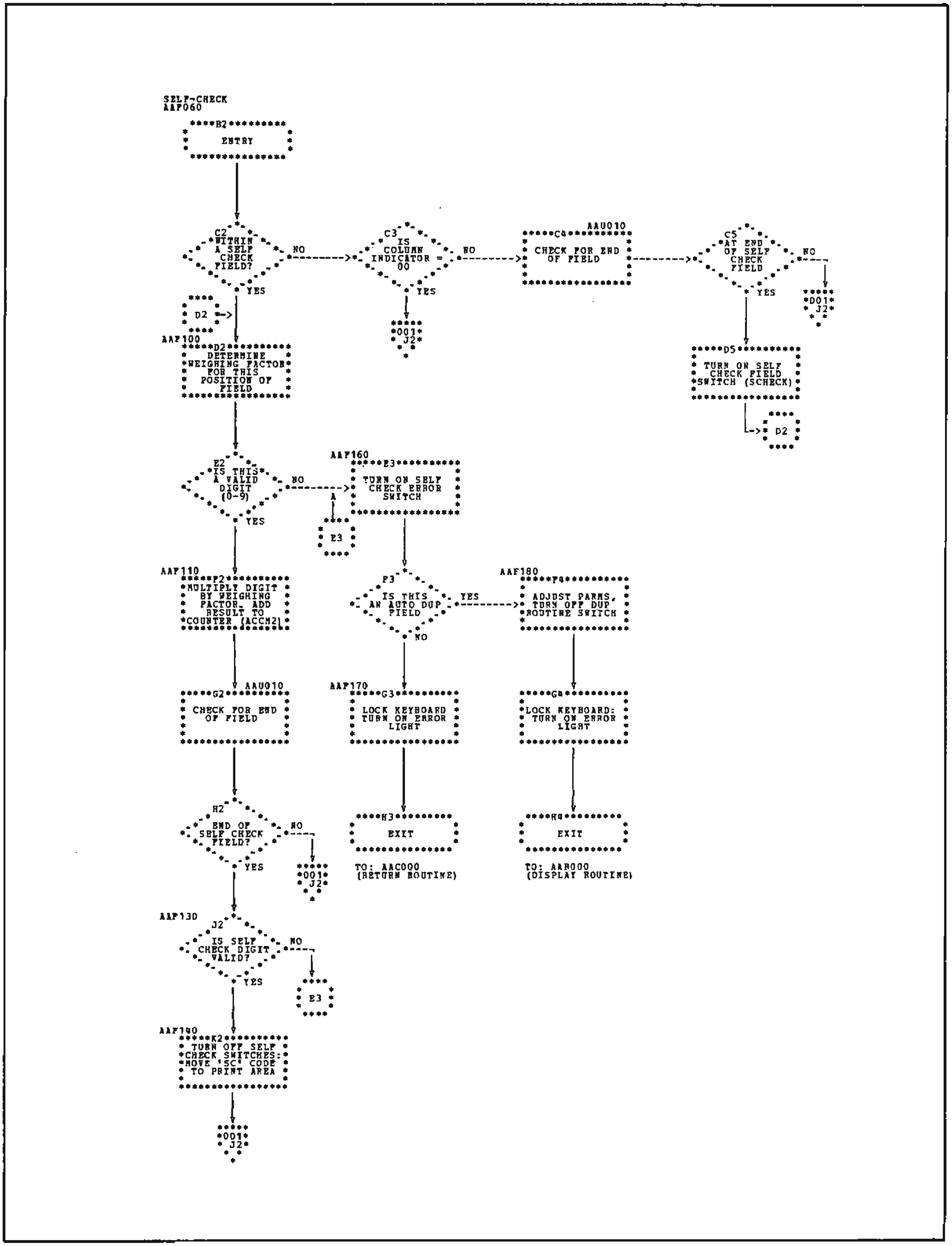

SELF-CHECK
AAF060
B2
ENTRY
C2
WITHIN A SELF CHECK FIELD?
NO
YES
C3
IS COLUMN INDICATOR = 00
NO
YES
001 J2
AAU010
C4
CHECK FOR END OF FIELD
C5
AT END OF SELF CHECK FIELD
NO
YES
001 J2
D2
AAF100
D2
DETERMINE WEIGHING FACTOR FOR THIS POSITION OF FIELD
D5
TURN ON SELF CHECK FIELD SWITCH (SCHECK)
D2
E2
IS THIS A VALID DIGIT (0-9)
NO
YES
AAF160
E3
TURN ON SELF CHECK ERROR SWITCH
E3
AAF110
F2
MULTIPLY DIGIT BY WEIGHING FACTOR. ADD RESULT TO COUNTER (ACCM2)
F3
IS THIS AN AUTO DUP FIELD
YES
NO
AAF180
F4
ADJUST PARMS, TURN OFF DUP ROUTINE SWITCH
AAU010
G2
CHECK FOR END OF FIELD
AAF170
G3
LOCK KEYBOARD TURN ON ERROR LIGHT
G4
LOCK KEYBOARD: TURN ON ERROR LIGHT
H2
END OF SELF CHECK FIELD?
NO
YES
001 J2
H3
EXIT
TO: AAC000 (RETURN ROUTINE)
H4
EXIT
TO: AAR000 (DISPLAY ROUTINE)
AAF130
J2
IS SELF CHECK DIGIT VALID?
NO
YES
E3
AAF140
K2
TURN OFF SELF CHECK SWITCHES: MOVE 'SC' CODE TO PRINT AREA
001 J2

Chart GA. Data Key/Self-Check Routine (Part 2 of 2)

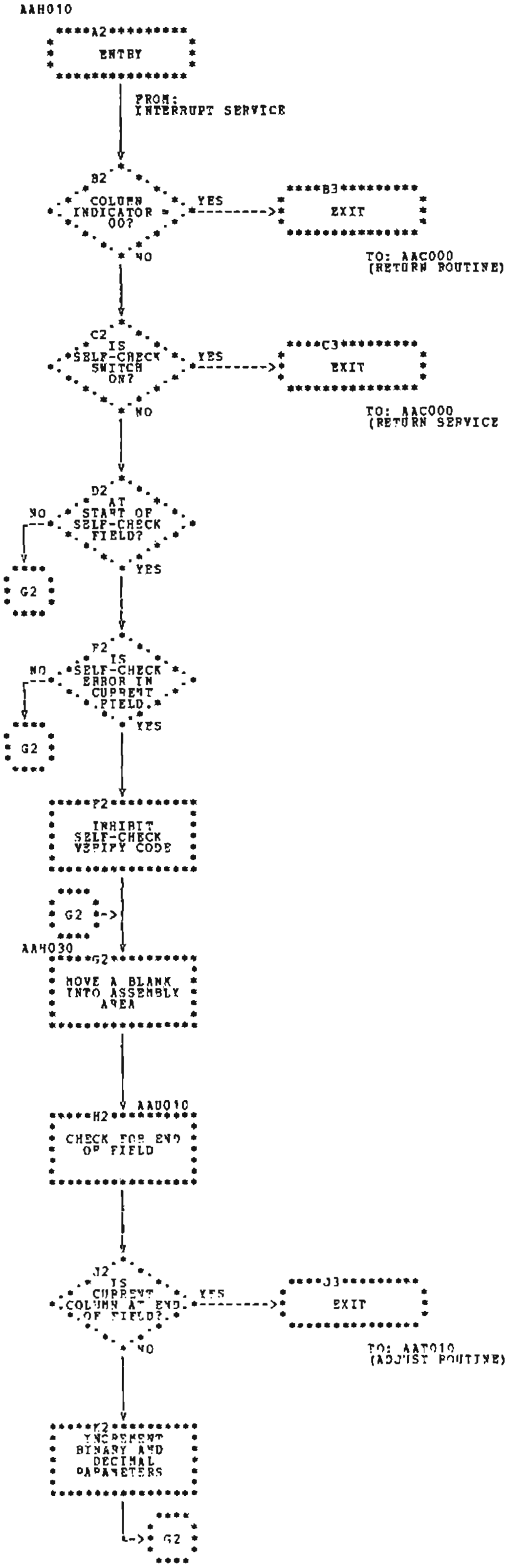


Chart GB. Skip Key Routine

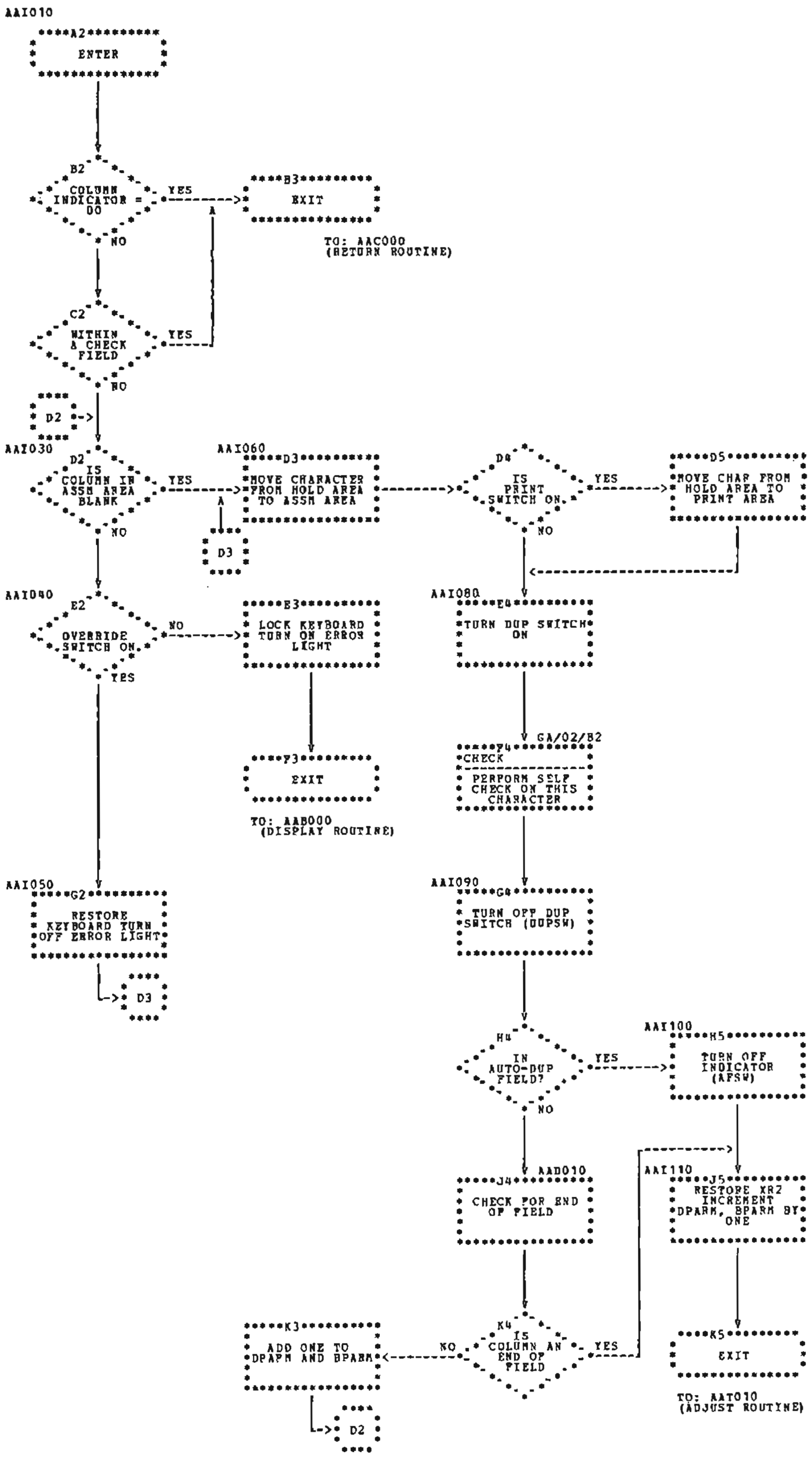

AAI010
A2
ENTER
B2
COLUMN INDICATOR = DO
YES
NO
B3
EXIT
TO: AAC000 (RETURN ROUTINE)
A
C2
WITHIN A CHECK FIELD
YES
NO
D2
AAI030
D2
IS COLUMN IN ASSM AREA BLANK
YES
NO
AAI060
D3
MOVE CHARACTER FROM HOLD AREA TO ASSM AREA
A
D3
D4
IS PRINT SWITCH ON.
YES
NO
D5
MOVE CHAR FROM HOLD AREA TO PRINT AREA
AAI040
E2
OVERRIDE SWITCH ON.
NO
YES
E3
LOCK KEYBOARD TURN ON ERROR LIGHT
AAI080
E4
TURN DUP SWITCH ON
F3
EXIT
TO: AAB000 (DISPLAY ROUTINE)
GA/02/B2
F4
CHECK
PERFORM SELF CHECK ON THIS CHARACTER
AAI050
G2
RESTORE KEYBOARD TURN OFF ERROR LIGHT
D3
AAI090
G4
TURN OFF DUP SWITCH (DUPSW)
H4
IN AUTO-DUP FIELD?
YES
NO
AAI100
H5
TURN OFF INDICATOR (AFSW)
AAD010
J4
CHECK FOR END OF FIELD
AAI110
J5
RESTORE XR2 INCREMENT DPARM, BPARM BY ONE
K3
ADD ONE TO DPARM AND BPARM
NO
K4
IS COLUMN AN END OF FIELD
YES
D2
K5
EXIT
TO: AAT010 (ADJUST ROUTINE)

Chart GC. Dup Key Routine

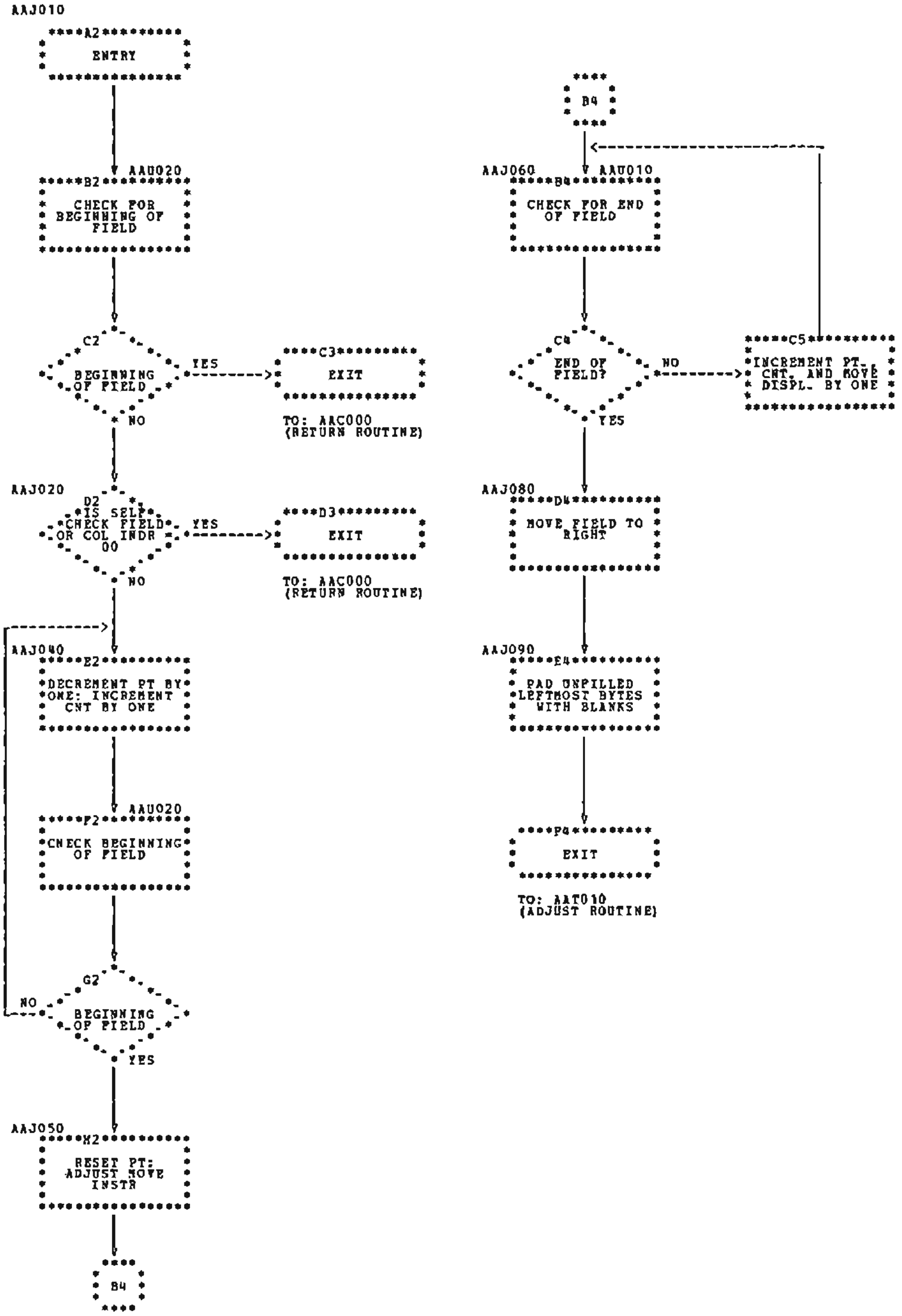

AAJ010
A2
ENTRY
AAU020
B2
CHECK FOR BEGINNING OF FIELD
C2
BEGINNING OF FIELD
YES
C3
EXIT
TO: AAC000 (RETURN ROUTINE)
NO
AAJ020
D2
IS SELF CHECK FIELD OR COL INDR 00
YES
D3
EXIT
TO: AAC000 (RETURN ROUTINE)
NO
AAJ040
E2
DECREMENT PT BY ONE: INCREMENT CNT BY ONE
AAU020
F2
CHECK BEGINNING OF FIELD
G2
NO
BEGINNING OF FIELD
YES
AAJ050
H2
RESET PT: ADJUST MOVE INSTR
B4
B4
AAJ060
AAU010
B4
CHECK FOR END OF FIELD
C4
END OF FIELD?
NO
C5
INCREMENT PT., CNT. AND MOVE DISPL. BY ONE
YES
AAJ080
D4
MOVE FIELD TO RIGHT
AAJ090
E4
PAD UNFILLED LEFTMOST BYTES WITH BLANKS
F4
EXIT
TO: AAT010 (ADJUST ROUTINE)

Chart GD. Right Adjust Key Routine

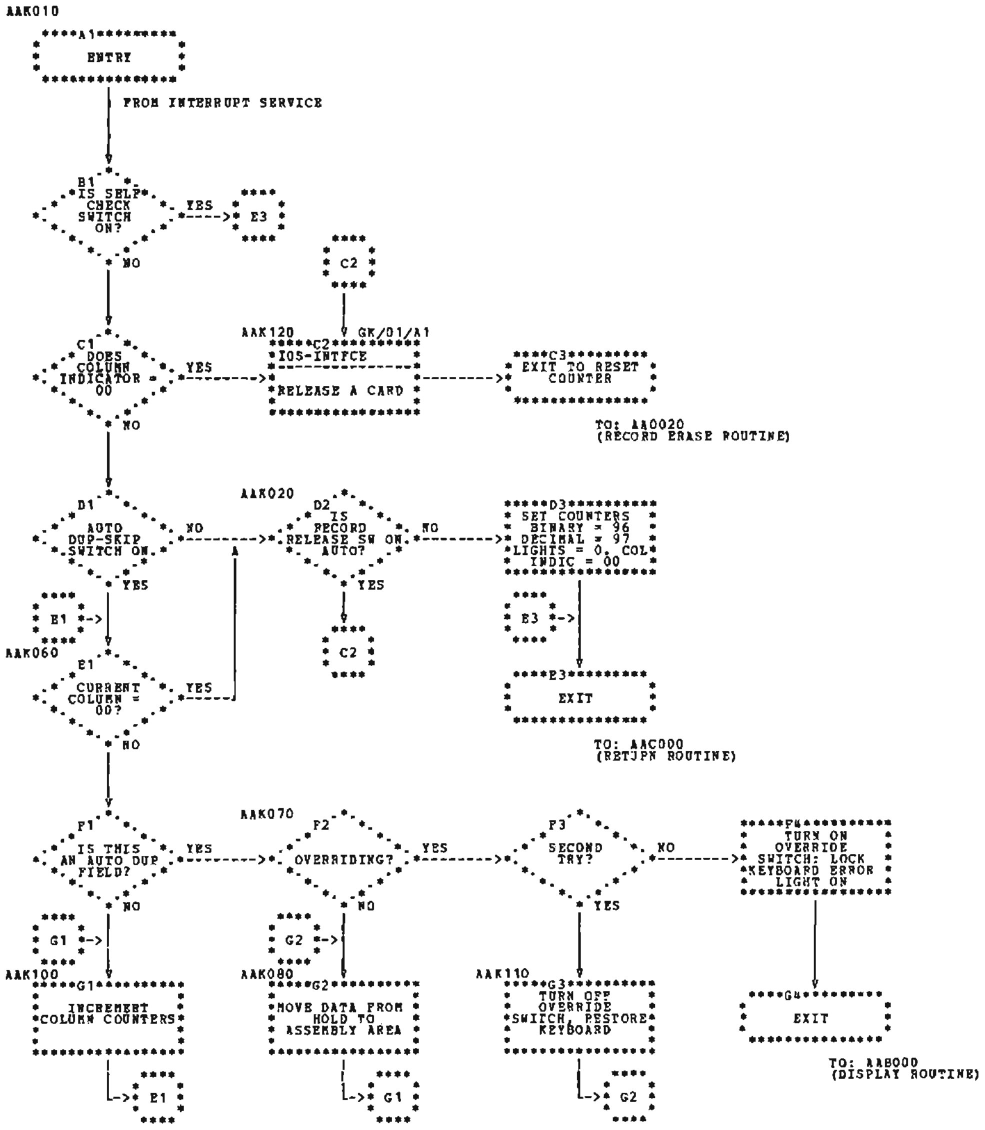

AAK010
A1
ENTRY
FROM INTERRUPT SERVICE
B1
IS SELF CHECK SWITCH ON?
YES
E3
NO
C2
AAK120
GK/01/A1
C1
DOES COLUMN INDICATOR = 00
YES
C2
IOS-INTFCE
RELEASE A CARD
C3
EXIT TO RESET COUNTER
TO: AA0020 (RECORD ERASE ROUTINE)
NO
AAK020
D1
AUTO DUP-SKIP SWITCH ON.
NO
A
D2
IS RECORD RELEASE SW ON. AUTO?
NO
D3
SET COUNTERS BINARY = 96 DECIMAL = 97 LIGHTS = 0, COL INDIC = 00
YES
E1
YES
C2
E3
AAK060
E1
CURRENT COLUMN = 00?
YES
E3
EXIT
TO: AAC000 (RETJPN ROUTINE)
NO
AAK070
F1
IS THIS AN AUTO DUP FIELD?
YES
F2
OVERRIDING?
YES
F3
SECOND TRY?
NO
F4
TURN ON OVERRIDE SWITCH: LOCK KEYBOARD ERROR LIGHT ON
NO
G1
NO
G2
YES
AAK100
G1
INCREMENT COLUMN COUNTERS
AAK080
G2
MOVE DATA FROM HOLD TO ASSEMBLY AREA
AAK110
G3
TURN OFF OVERRIDE SWITCH, RESTORE KEYBOARD
G4
EXIT
TO: AAB000 (DISPLAY ROUTINE)
E1
G1
G2

Chart GE. Release Key Routine

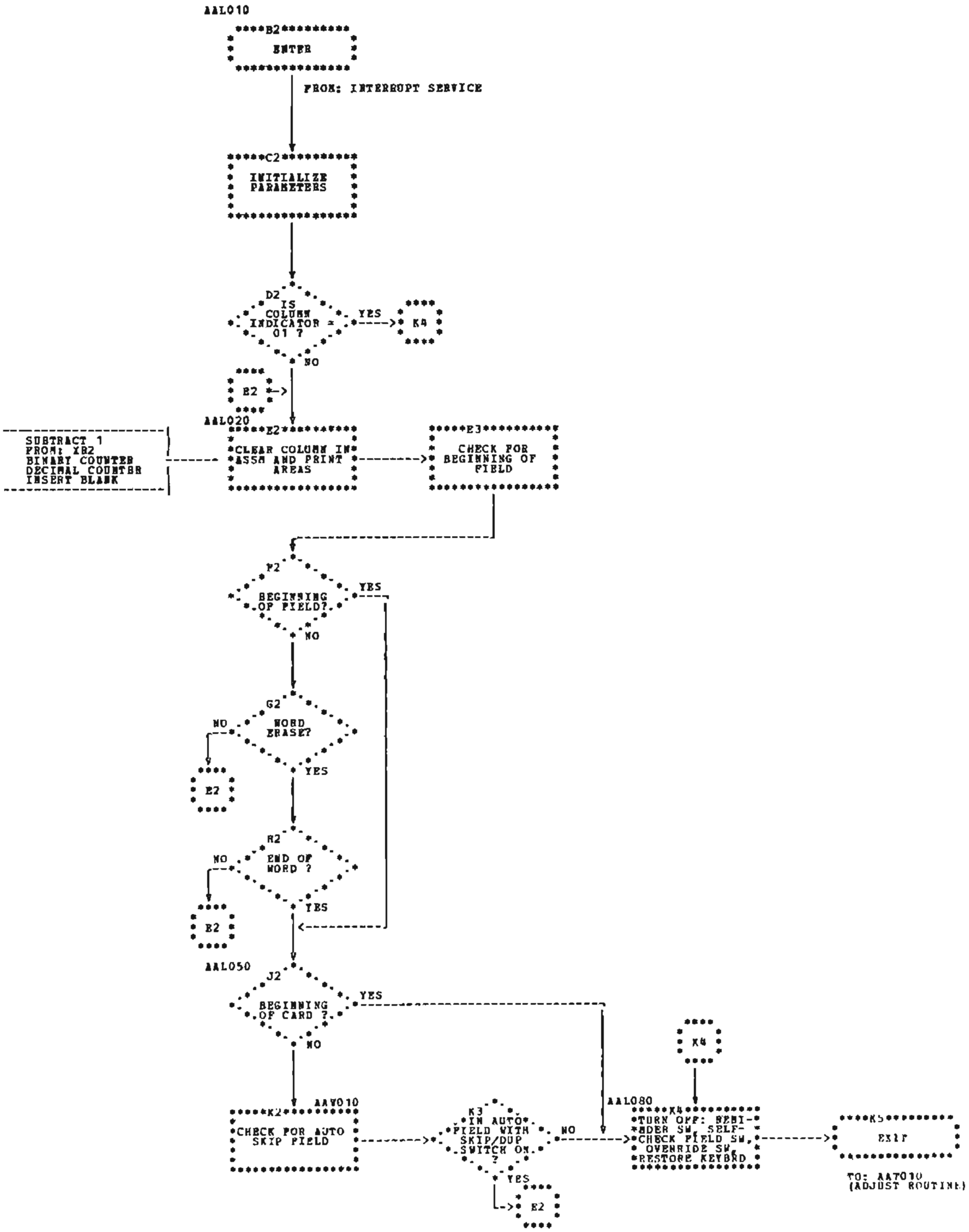

AAL010
B2
ENTER
FROM: INTERRUPT SERVICE
C2
INITIALIZE PARAMETERS
D2
IS COLUMN INDICATOR = 01 ?
YES
K4
NO
E2
AAL020
E2
SUBTRACT 1 FROM: XB2 BINARY COUNTER DECIMAL COUNTER INSERT BLANK
CLEAR COLUMN IN ASSM AND PRINT AREAS
E3
CHECK FOR BEGINNING OF FIELD
F2
BEGINNING OF FIELD?
YES
NO
G2
WORD ERASE?
NO
E2
YES
H2
END OF WORD ?
NO
E2
YES
AAL050
J2
BEGINNING OF CARD ?
YES
NO
K4
AAV010
K2
CHECK FOR AUTO SKIP FIELD
K3
IN AUTO FIELD WITH SKIP/DUP SWITCH ON. ?
NO
YES
E2
AAL080
K4
TURN OFF: REMI-NDER SW, SELF-CHECK FIELD SW, OVERRIDE SW, RESTORE KEYBRD
K5
EXIT
TO: AAT010 (ADJUST ROUTINE)

Chart GF. Field Erase Key Routine

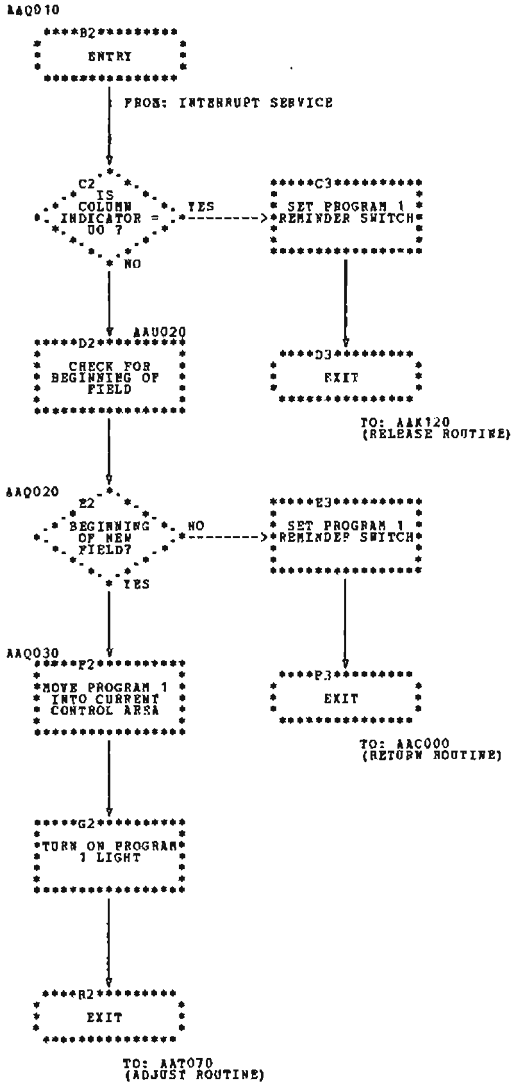

AAQ010
B2
ENTRY
FROM: INTERRUPT SERVICE
C2
IS COLUMN INDICATOR = 00 ?
YES
C3
SET PROGRAM 1 REMINDER SWITCH
NO
AAU020
D2
CHECK FOR BEGINNING OF FIELD
D3
EXIT
TO: AAK120 (RELEASE ROUTINE)
AAQ020
E2
BEGINNING OF NEW FIELD?
NO
E3
SET PROGRAM 1 REMINDER SWITCH
YES
AAQ030
F2
MOVE PROGRAM 1 INTO CURRENT CONTROL AREA
F3
EXIT
TO: AAC000 (RETURN ROUTINE)
G2
TURN ON PROGRAM 1 LIGHT
H2
EXIT
TO: AAT070 (ADJUST ROUTINE)

Chart GG. Program 1 Key Routine

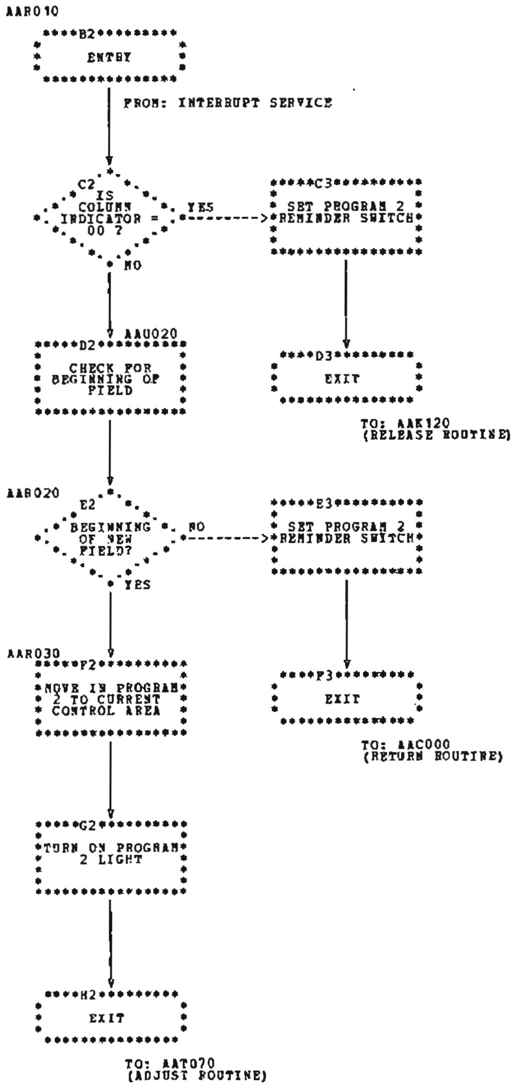
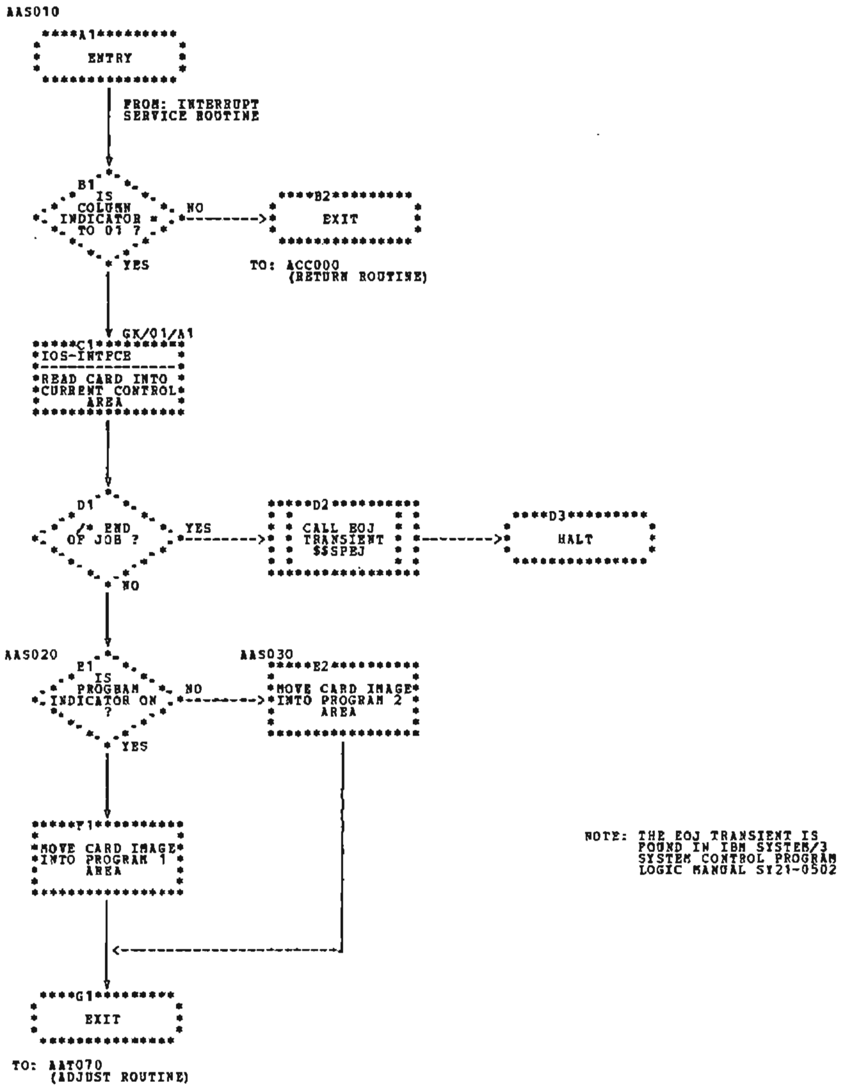

AAB010
B2
ENTRY
FROM: INTERRUPT SERVICE
C2
IS COLUMN INDICATOR = 00 ?
YES
NO
C3
SET PROGRAM 2 REMINDER SWITCH
AAU020
D2
CHECK FOR BEGINNING OF FIELD
D3
EXIT
TO: AAK120 (RELEASE ROUTINE)
AAB020
E2
BEGINNING OF NEW FIELD?
NO
YES
E3
SET PROGRAM 2 REMINDER SWITCH
AAB030
F2
MOVE IN PROGRAM 2 TO CURRENT CONTROL AREA
F3
EXIT
TO: AAC000 (RETURN ROUTINE)
G2
TURN ON PROGRAM 2 LIGHT
H2
EXIT
TO: AAT070 (ADJUST ROUTINE)

Chart GH. Program 2 Key Routine


AAS010
A1
ENTRY
FROM: INTERRUPT SERVICE ROUTINE
B1
IS COLUMN INDICATOR TO 01 ?
NO
B2
EXIT
TO: ACC000 (RETURN ROUTINE)
YES
GK/01/A1
C1
IOS-INTFCE
READ CARD INTO CURRENT CONTROL AREA
D1
/* END OF JOB ?
YES
D2
CALL EOJ TRANSIENT $$SPEJ
D3
HALT
NO
AAS020
E1
IS PROGRAM INDICATOR ON ?
NO
AAS030
E2
MOVE CARD IMAGE INTO PROGRAM 2 AREA
YES
F1
MOVE CARD IMAGE INTO PROGRAM 1 AREA
G1
EXIT
TO: AAT070 (ADJUST ROUTINE)

NOTE: THE EOJ TRANSIENT IS FOUND IN IBM SYSTEM/3 SYSTEM CONTROL PROGRAM LOGIC MANUAL SY21-0502

Chart GI. Program Load Switch Routine

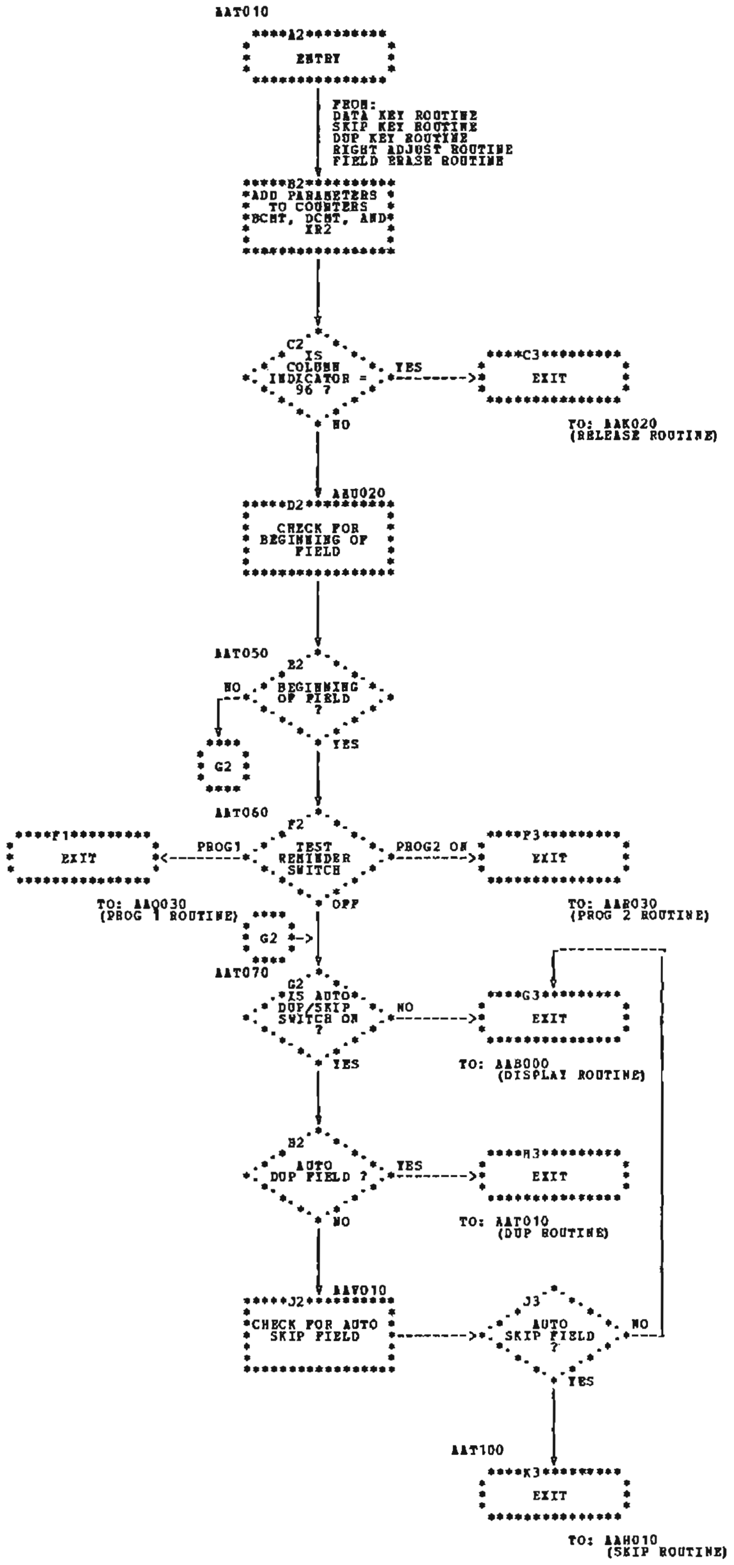


Chart GJ. Adjust Routine

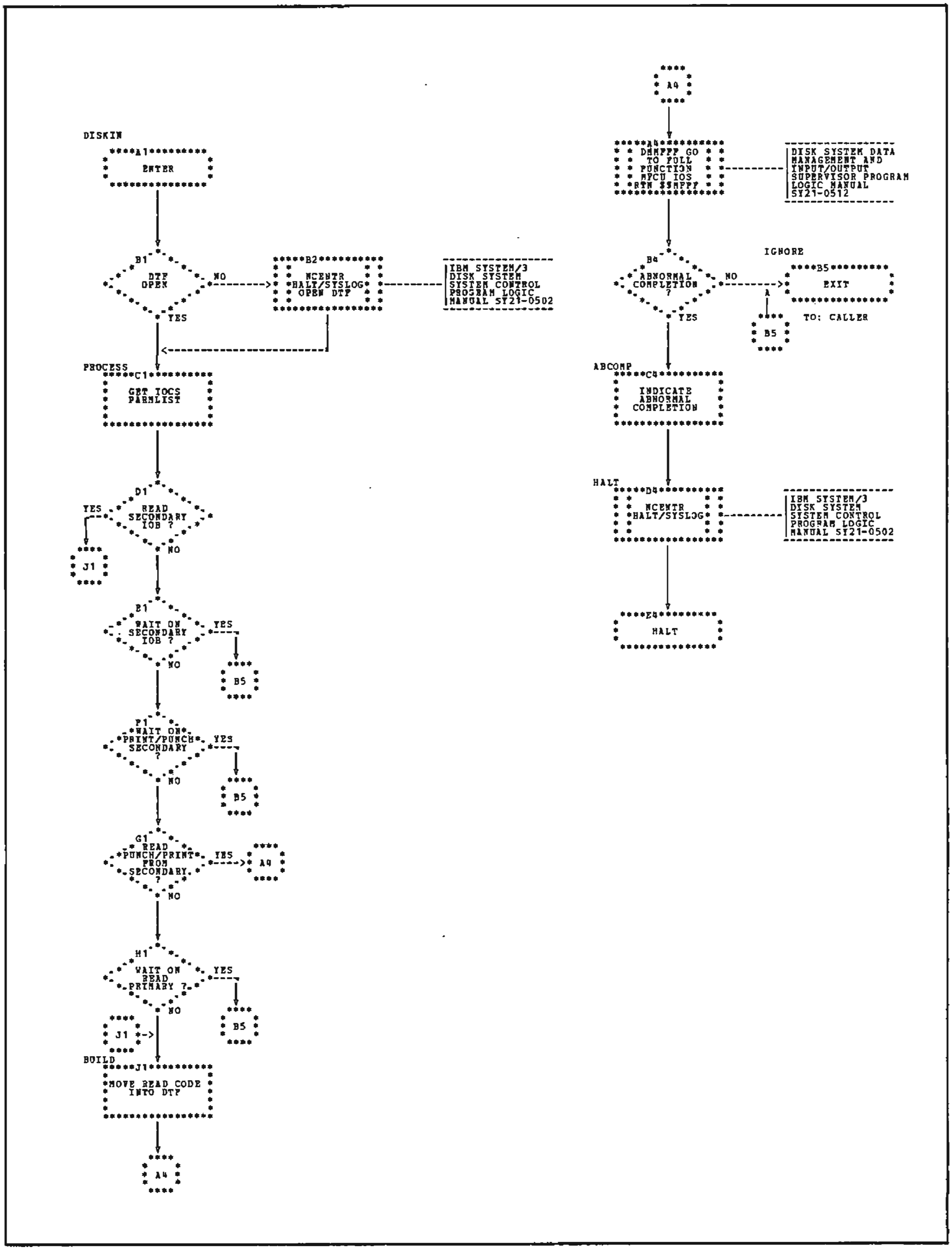

DISKIN
A1
ENTER
B1
DTF OPEN
NO
YES
B2
NCENTR HALT/SYSLOG OPEN DTF
IBM SYSTEM/3 DISK SYSTEM SYSTEM CONTROL PROGRAM LOGIC MANUAL SY21-0502
PROCESS
C1
GET IOCS PARMLIST
D1
READ SECONDARY IOB ?
YES
NO
J1
E1
WAIT ON SECONDARY IOB ?
YES
NO
B5
F1
WAIT ON PRINT/PUNCH SECONDARY ?
YES
NO
B5
G1
READ PUNCH/PRINT FROM SECONDARY ?
YES
NO
A4
H1
WAIT ON READ PRIMARY ?
YES
NO
B5
J1
BUILD
J1
MOVE READ CODE INTO DTF
A4
A4
A4
DMMPFF GO TO POLL FUNCTION MPCU IOS RTN $$MPFF
DISK SYSTEM DATA MANAGEMENT AND INPUT/OUTPUT SUPERVISOR PROGRAM LOGIC MANUAL SY21-0512
B4
ABNORMAL COMPLETION ?
NO
YES
IGNORE
B5
EXIT
TO: CALLER
B5
ABCOMP
C4
INDICATE ABNORMAL COMPLETION
HALT
D4
NCENTR HALT/SYSLOG
IBM SYSTEM/3 DISK SYSTEM SYSTEM CONTROL PROGRAM LOGIC MANUAL SY21-0502
E4
HALT

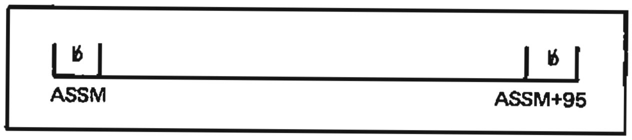
Chart GK. IOS Interface Routine

Section 4. Data Area Formats

This section describes data areas that are used by more than two routines.

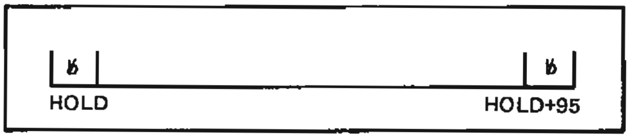
Assembly Area – ASSM

This 96-byte area is initially filled with blanks. It is used as a read buffer for both hoppers and as a work area for building the card image. This area is aligned on a hexadecimal 80 boundary.



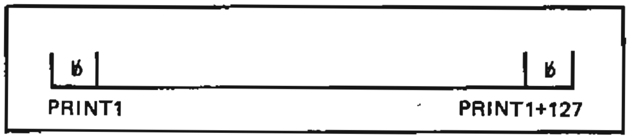
Hold Area -- HOLD

This 96-byte area is initially filled with blanks. After the first card is punched, this area is used to hold the image of the last card released. The hold area is also used as the punch buffer. This area is aligned on a hexadecimal 80 boundary.



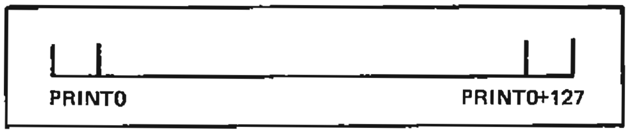
Print Area -- PRINT1

This 128-byte area is initially filled with blanks. The PRINT1 area is used as a work area for building the print buffer.



Print Area -- PRINT0

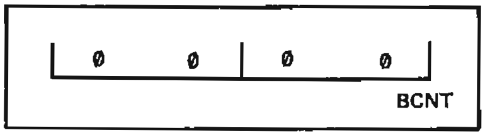
This 128-byte area is not initialized. It is used as the print buffer and is aligned on a X'80' boundary.



Binary Column Indicator -- BCNT

This 2-byte area is initially set to zero. This indicator is used to index through the card work areas on a column by column basis. The setting of this indicator reflects the number of card columns that have been built in the work areas. The possible range is from 0 to 96 (decimal values).

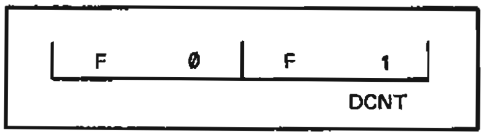
Note: Data is in a binary format.

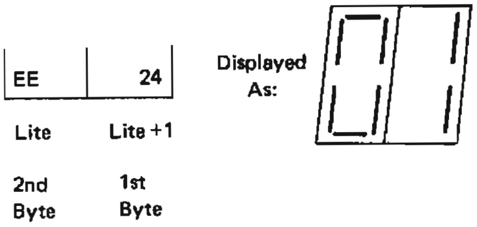


Decimal Column Indicator -- DCNT

This 2-byte area is initially set to 01 (decimal value). This indicator is used to reflect the column that is being operated on. The value of DCNT is used in selecting the respective light combination displayed on the column indicator on the front of the Data Entry Keyboard.

Note: Data is in a zoned decimal format.





	Bit 0	Bit 1	Bit 2	Bit 3	Bit 4	Bit 5	Bit 6	Bit 7
1st Byte (Unit's)	Lighted Segment E	Lighted Segment D	Lighted Segment F	Lighted Segment C	Lighted Segment B	Lighted Segment G	Lighted Segment A	Prog 2 Indicator
2nd Byte (Ten's)	Lighted Segment E	Lighted Segment D	Lighted Segment F	Lighted Segment C	Lighted Segment B	Lighted Segment G	Lighted Segment A	Prog 1 Indicator

Hex Code	Digit	Lighted Segments
EE	0	ABDEFG
24	1	FG
BA	2	ABCEF
B6	3	ACEFG
74	4	CDFG
D6	5	ACDEG
DE	6	ABCDEG
A4	7	EFG
FE	8	ABCDEFG
F6	9	ACDEFG

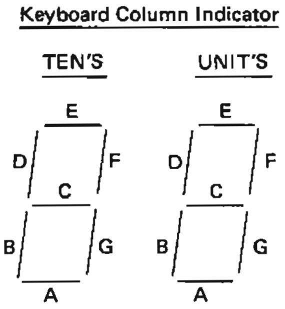


Figure 5-4. Column Indicator Data Area and Display Values

Column Indicator – LITE

This is a 2-byte area, initially set to X'EE', X'24'. This area is used as an input area by the LIO instruction that displays the column indicator. This indicator is initially displayed as 01 (see Figure 5-4).

Stick-Light Table -- TAB

This is a 10-byte area consisting of hexadecimal values for the decimal numbers 0-9. The decimal counter (DCNT) is used in referencing this table (see Figure 5-5). The entries from this table (TAB) are placed in LITE.

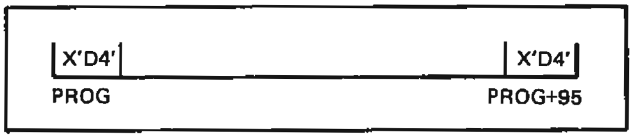
X'EE'	X'24'	X'BA'	X'B6'	X'74'	X'D6'	X'DE'	X'A4'	X'FE'	X'F6'
TAB									TAB+9

Hex Byte Value	Displayed on Keyboard as:
X'EE'	0
X'24'	1
X'BA'	2
X'B6'	3
X'74'	4
X'D6'	5
X'DE'	6
X'A4'	7
X'FE'	8
X'F6'	9

Figure 5-5. Stick Light Table and Display Values

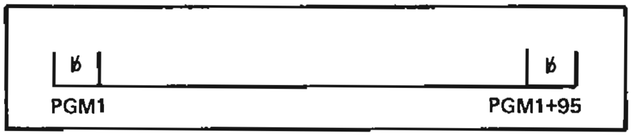
Current Control Area -- PROG

This is a 96-byte area in which all bytes are initially set to X'D4' (code for end of field and lower shift). When the program is under manual control, all bytes of this area are set to X'D4'. This area always contains the image of the program card that is in current control.



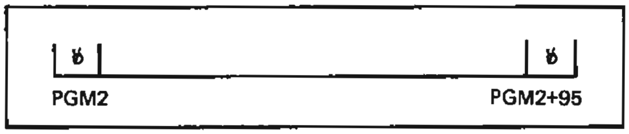
Program 1 -- PGM1

This is a 96-byte storage area used to contain the image of the program 1 control card. This area is initially filled with blanks.



Program 2 -- PGM2

This is a 96-byte storage area used to contain the image of the program 2 control card. This area is initially filled with blanks.

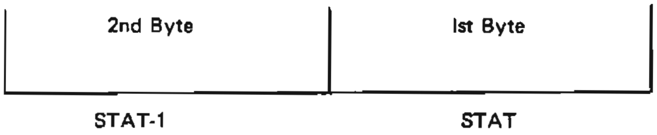


Sense Table -- STAT/DATA

This is a 4-byte area used in detecting the source of an interrupt. The first two bytes are referenced by the label STAT and indicate whether a function key was the cause of the interrupt (see Figure 5-6). The second two bytes are referenced by the label DATA and indicated whether a data key was the source of the interrupt (see Figure 5-7). The sense table receives the sense data from the SNS instructions to the Data Entry Keyboard.

Control Code -- CCODE

This is a 1-byte area used as the Q code for the SIO instructions in the Return routine. This area is initially set to hexadecimal 0F (see Figure 5-8).



	Bit 0	Bit 1	Bit 2	Bit 3	Bit 4	Bit 5	Bit 6	Bit 7
Ist Byte	AUTO SK/ DUP Switch	RECORD ERASE Switch	Reserved	PROG Switch	SKIP Key	DUP Key	AUTO REC REL Switch	Function Key Interrupt
2nd Byte	PROG 1 Key	PROG 2 Key	PROG LOAD Switch	REL Key	ERASE Key	ERROR RESET Key	READ Key	RIGHT ADJUST Key

Figure 5-6. Function Key Sense Table

2nd Byte	Ist Byte
DATA-1	DATA

	Bit 0	Bit 1	Bit 2	Bit 3	Bit 4	Bit 5	Bit 6	Bit 7
Ist Byte	PRINT Switch	Reserved	LOWER SHIFT Key	Invalid Character*	Reserved	MULT PCH Key	Reserved	Data Key Interrupt
2nd Byte	Data Character Keyed (EBCDIC)							

* The presence of this bit indicates that numeric shift was programmed and a character other than 0 through 9 or space was keyed.

NOTE: If the bit = 1, that key or switch is on.
If the bit = 0, that key or switch is off.

Figure 5-7. Data Key Sense Table

X'OF'

CCODE

Control Code Bits

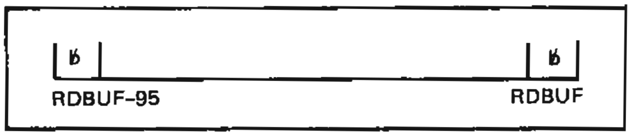
Bit 0–Programmed Numeric Mode
Bit 1–Programmed Lower Shift
Bit 2–Error Indicator
Bit 3–Spare
Bit 4–Restore Data Key
Bit 5–Unlock Data Key
Bit 6–Enable/Disable Interrupt
Bit 7–Reset Interrupt

Note: If bit = 1, that function is enabled.
If bit = 0, that function is disabled.

Figure 5-8. Control Code Data Area

Read Buffer -- RDBUF

A card to be punched is read into the assembly area (ASSM) from the secondary hopper; the image of that card is moved to a 96-byte area called RDBUF. The RDBUF is used as a compare area when checking the override feature.



Copyright

This 46-byte area contains the program number for this program and copyright information as follows: 5702-UT1ƀCOPYRIGHTƀIBMƀCORPƀ1970. (The ƀ represents a blank). The remainder of the area is filled with blanks.

MFCU IOCS Parameter List

These areas consist of six parameters, each 6 bytes long. They contain the MFCU IOCS operation code and stacker select information. The codes and information are passed to the IOS Interface routine which places an equivalent of them in the DTF. The codes are shown in Figure 5-9.

Buffer-Associated IOB -- PUIOB, RDIOB

MIODAT: A 2-byte area that contains the address of the buffer associated with this IOB.

MIOFLG: A 1-byte area reserved for the completion code.

MIODCH: A 2-byte area containing a pointer to the next buffer-associated IOB where more than one I/O buffer is used.

Define the File – DTF

This is a 36-byte parameter list which is passed to the Full Function MFCU IOS routine ($$MFFF). It contains operation codes to indicate which MFCU functions are to be performed. Figure 5-10 shows the DTF parameter list.

NAME	CODE	DESCRIPTION
RPNPRS	X'870000800000'	Read, punch, print, from secondary IOB
WRDP	X'100000800000'	Wait on read primary IOB
RDP	X'900000800000'	Read primary IOB
RDS	X'810000800000'	Read secondary IOB
WRDS	X'010000800000'	Wait on read secondary IOB
WPNPRS	X'060000800000'	Wait on punch, print from secondary IOB

Note: The wait codes, WRDP, WRDS, and WPNPRS are ignored because the Full Function MFCU IOS routine automatically handles them before performing an operation such as read or print.

Figure 5-9. IOCS Parameter List

NAME	BYTE	BIT NUMBER	DESCRIPTION
MDFDEV	0		Device Address
MDFUPS	1		UPSI Mask for this DIF
			First Attribute Byte
MDFAT1	2	0	Indicates input
		1	Indicates Output
			Second Attribute Byte
MDFAT2	3	3	Indicates dual I/O area
		5	Hopper used as system input device
		6	1 read on last input operation
		7	File is open
MDFCHA	4-5		DTF backward chaining address
MDFCHB	6-7		DTF forward chaining address
MDFARR	8-9		ARR save area
MDFXR1	10-11		XR1 save area
MDFLRA	12-13		Logical record address
MDFCMP	14		Completion code
			Operation Code
MDFOPP	15	0	READ
		1	PRINT
		2	PUNCH
		3	MOVE
MDFSTS	16		Stacker select parameter byte
MDFQ	17		Q code-device address same as byte 0
	18-22		Work area
MDFSUA	23-24		Address of permanent save area
MDFERP	25-26		Disk address of Error Recovery program
MDFRIO	27-28		Address of current read IOB
MDFVIO	29-30		Address of current punch IOB
MDFPUB	31-32		Address of current punch I/O area
MDFPTB	33-34		Address of print I/O area
MDFPTL	35		Print buffer length
MDFPUL	36		Punch buffer length

Figure 5-10. DTF Diagram

Data Area Activity

Figure 5-11 shows which Data Recording routines use the data areas described in this section.

Using Routines / Data Areas	ASSM	HOLD	PRINT0	PRINT1	PROG	PGM1	PGM2	DATA/STAT	CCODE	BCNT	DCNT	RDBUF	LITE	TAB	IOCS Parameter List	DTF
IOS Interface Routine	X	X													X	X
Initializing Routine	X							X				X				
Display Routine											X		X	X		
Return Routine					X				X				X			
Interrupt Handler								X								
Interrupt Service					X			X					X			
Data Key Routine	X			X	X			X	X	X	X	X				
Invalid Character Routine									X							
Skip Key Routine	X				X					X						
Dup Key Routine	X	X		X				X	X	X		X				
Right Adjust Key Routine	X			X						X						
Release Key Routine	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	
Field Erase Key Routine	X			X	X			X	X	X						
Error Reset Key									X	X						
Record Release Switch Routine										X						
Record Erase Switch Routine	X			X					X	X	X					
Read Key Routine		X								X					X	
Program 1 Key Routine					X	X			X	X			X			
Program 2 Key Routine					X		X		X	X			X			
Program Load Switch Routine	X				X	X	X			X			X		X	
Adjust Routine					X			X		X	X					

Figure 5-11. Data Area Activity Chart

Part VI. Data Verifying

Section 1. Introduction

The IBM System/3 Data Verifying program causes the IBM System/3 to function as if it were the IBM 5496 Data Recorder operating in the verify mode. In simulating the Data Recorder, this program accepts control cards which specify the format of the card image during card verification.

System Requirements

The IBM System/3 Data Verifying program operates using the following system configurations:

- The IBM 5410 Processing Unit.
- The IBM 5424 Multi-Function Card Unit (MFCU).
- The IBM 5475 Data Entry Keyboard.
- The IBM 5444 Disk Storage Drive.

Section 2. Method of Operation

This section is concerned with the functional flow of logic and data for the System/3 Data Verifying program. The following section, *Section 3, Program Organization,* will expand upon the items found in this functional overview. This section consists of diagrams and supporting text.

General Flow of the Data Verifying Program

Upon loading the Data Verifying program, the Initializing routine senses the Data Entry Keyboard status into the sense table (SNS instruction). Data areas are then cleared or initialized to a predetermined setting (column indicator set to 01). The column indicator is displayed, and control is returned to the Return routine to wait for an interrupt from the Data Entry Keyboard.

When an interrupt is detected (see Figure 6-1), control is passed to the Interrupt Handler routine which disables any further interrupts until the current interrupt is resolved. The keyboard is then sensed for its current status, and this status is then sensed into the sense table. Control is then passed to the Interrupt Service routine which tests the sense table. The source of the interrupt is determined, and control is passed to the respective routine to service the interrupt.

Four general types of actions can result, depending upon the type of interrupt (see Figure 6-1):

- Type A: The assembly area (ASSM) is placed under program control.

- Type B: The format of the card image (in the assembly area) is modified in accordance with the program control card entries.

- Type C: A character is compared against the card image in the assembly area. If the characters are the same, that column is correct; otherwise an error condition occurs. After the third consecutive error condition, the character replaces the corresponding character in the assembly area.

- Type D: The verified card is printed with an OK in columns 127-128. If the original card was not correct, the corrected card is punched and printed (without OK) and is inserted into the verified deck. The incorrect card is stacker selected into another stacker.

After an interrupt is serviced, control is passed back to the Return routine. The Return routine then waits for the next interrupt to occur.

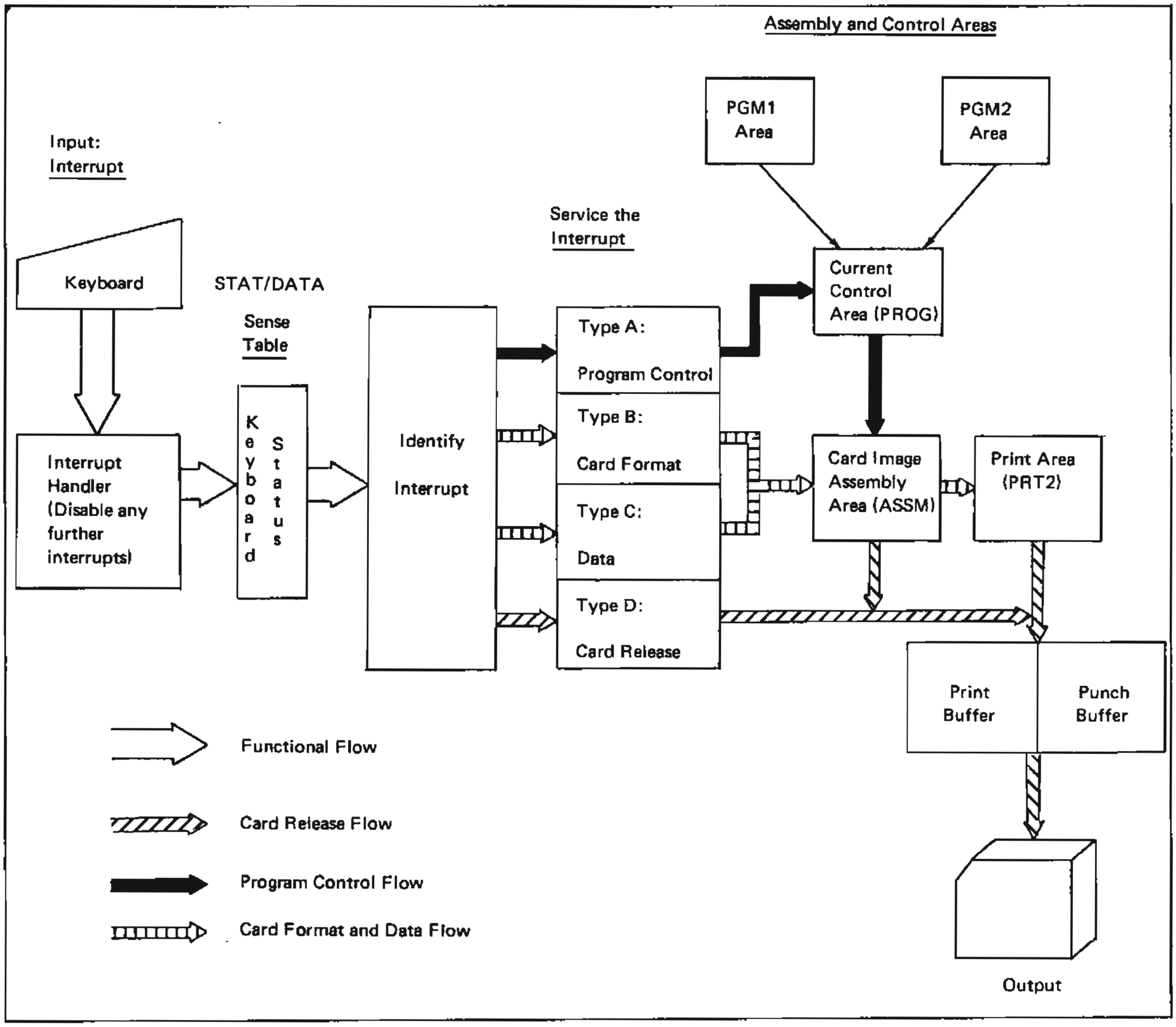


Figure 6-1. Functional Flow of Data and Control for Data Verifying Program

Section 3. Program Organization

This section is designed to show how the routines that comprise the Data Verifying program are interconnected. Figure 6-2 shows the general layout of the separate routines. The text that follows the figure explains the function of each routine. (Flowcharts are included for routines that are complex.)

Figure 6-3 shows a storage map showing the order that data areas and routines reside in the Data Verifying program.

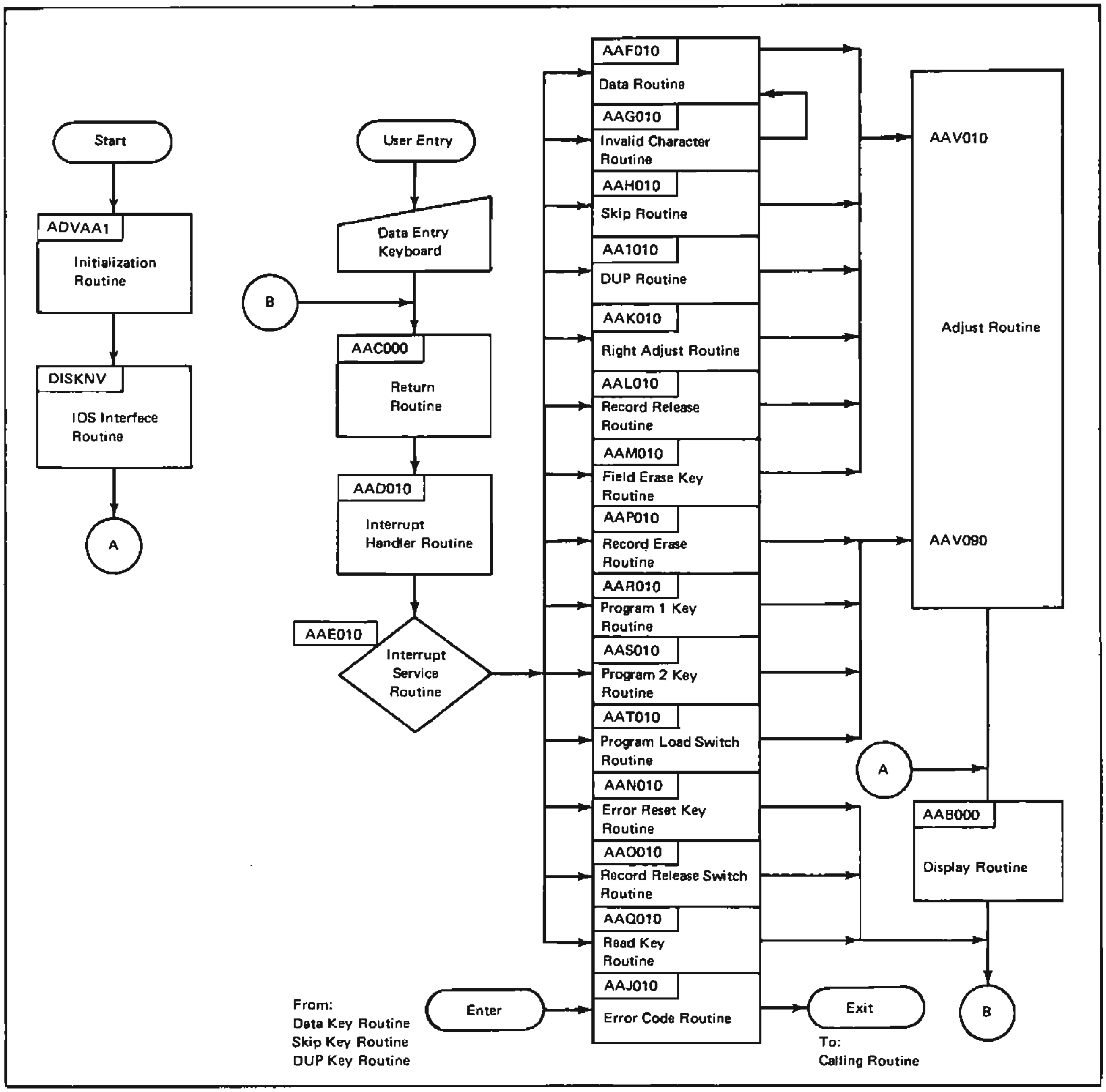


Figure 6-2. Program Organization of Data Verifying Program

ART: 55200

Supervisor (System Communication Area)
ASSM (Assembly Area)
PRINT1 (Print Buffer)
Copyright
PROG and Constants
HOLD (Hold Area)
CORRT (Correct Area)
PGM1, PGM2
Initializing Routine
Column Indicator Display Routine
Return Routine
Interrupt Handler Routine
Interrupt Service Routine
Data Key Routine
Invalid Character Routine
Skip Key Routine
DUP Key Routine
Error Code Routine
Right Adjust Key Routine
Release Key Routine
Field Erase Key Routine
Error Reset Key Routine
Record Release Switch Routine
Record Erase Switch Routine
Read Key Routine
Program 1 Key Routine
Program 2 Key Routine
Program Load Switch Routine
End or Beginning of Field Routine
Adjust Routine
Test for Auto Skip Field Routine
IOS Interface Routine
Full Function MFCU IOS Routine

Figure 6-3. Storage Map for Data Verifying Program

Initializing Routine

Entry Point: ADVAA1

Chart: None

Function:

- Calls the Program Protect transient routine which checks for copyright violation.
- Loads register 1 with base address.
- Loads the level 1 instruction address register (IAR) with address of Interrupt Handler routine.
- Loads register 2 with base address.
- Senses the keyboard status into the sense table.

Column Indicator Display Routine

Entry Point: AAB000

Chart: None

Function:

- Loads the LITE data area for the Return routine by indexing TAB data area with the decimal value of DCNT.

Return Routine

Entry Point: AAC000

Chart: None

Function:

- Displays column indicator (LITE) data area.
- Displays on-off status of program 1 and/or program 2.
- Completes current interrupt and enables further interrupts by giving a SNS instruction.
- Waits for interrupt to occur.
- Passes control to Interrupt Handler routine (AAD010) when an interrupt occurs.
- Passes control to the Interrupt Service routine (AAE010) after returning from the Interrupt Handler routine.

Interrupt Handler Routine

Entry Point: AAD010

Chart: None

Function: When an interrupt occurs, there is a physical exchange in the IAR instruction address register and the level 1 IAR instruction address register giving control to the Interrupt Handler routine. The Interrupt Handler routine performs the following:

- Disables any further interrupts by use of an SIO instruction.
- Senses the keyboard status into the sense table.
- Returns control to the Return routine.

Interrupt Service Routine

Entry Point: AAE010

Chart: None

Function: Upon receiving control from the Return routine, this routine:

- Tests sense table to locate the source of the interrupt.
- Passes control to the appropriate routine.
- Initiates halt if no source is found or if an invalid interrupt is detected.

Data Key Routine

Entry Point: AAF010

Chart: HA

Function:

- Compares the character just keyed with character in corresponding card column in the assembly area (ASSM).
- If comparison was equal, branches to Adjust routine (AAV010).
- Locks keyboard and turns on error light if above comparison was unequal.
- Enters keyed character into assembly area (ASSM) on third try.
- Enters correct data into print work area (PRT2), if PRINT switch is on.

Invalid Character Routine

Entry Point: AAG010

Chart: None

Function: If an invalid character is entered from the Data Entry Keyboard:

- The Data Entry Keyboard is locked.
- The error light is turned on.

Note: An invalid character occurs when the current column was programmed for a numeric shift and a character other than 0-9 or blank was keyed.

Skip Key Routine

Entry Point: AAH010

Chart: HB

Function:

- Compares assembly area (ASSM) to blank.
- If comparison was equal and the current column was at the end of the field, branch to Adjust routine (AAV010).
- Locks keyboard and turns on error light if above compare is unequal.
- Enters a blank into assembly area (ASSM) on third try.

DUP Key Routine

Entry Point: AAI010

Chart: HC

Function:

- Compares hold area with assembly area (ASSM).
- If comparison was equal, branches to Adjust routine (AAV010).
- Locks keyboard, turns on error light if above compare is unequal.
- After third try, moves in character from hold area to assembly area (ASSM).
- Enters correct data into print work area (PRT2) from hold area if PRINT switch is on.

Error Code Routine

Entry Point: AAJ010

Chart: None

Function:

- Turns on reverify switch.
- Restores error count to zero.
- Turns on the error code in the corresponding column of Tier 4 in the print work area (PRT2). (See *Section 4. Data Area Formats, Print 2 Area – PRT2* for error codes.)

Right Adjust Key Routine

Entry Point: AAK010

Chart: HD

Function:

- Checks for end of right adjust field.
- Resets keyboard; turns off error light.
- Resets all right adjust switches.
- Checks that all positions in field have been compared.
- Branches to Adjust routine (AAV010).

Release Key Routine

Entry Point: AAL010

Chart: HE (parts 1, 2, 3, and 4)

Function:

- This routine moves the column indicator up through column 96 to column 00 under the following conditions:

  1. If the column is programmed for automatic duplication and the AUTO SK/DUP switch is on, the data in the assembly area is compared to the data in the hold area. (See *Section 3. Program Organization, DUP Key Routine* for error conditions).

  2. If the column is programmed for automatic skipping the AUTO SK/DUP switch is on, the column is skipped over.

  3. In all other cases, the corresponding column in the assembly area (ASSM) is compared to a blank. (See *Section 3. Program Organization, Skip Key Routine* for error conditions.)

- Prints OK on verified card.
- Punches and prints a corrected card if necessary.
- Prints error codes on incorrect card.
- Reads next card to be verified.

Field Erase Key Routine

Entry Point: AAM010

Chart: HF

Function:

- Causes the column indicator to backspace to:

  1. The beginning of the last manual field.

  2. The beginning of the last keyed word.

- Restores original card information into assembly area (ASSM) on a field basis. Blanks out corresponding area in print work area (PRT2).
- Resets Remember switch to zero (this disallows any program level changes).
- Resets the error code bit in the print work area (PRT2) for the corresponding column that was erased.

Error Reset Key Routine

Entry Point: AAN010

Chart: None

Function:

- Restores operational functions of the Data Entry Keyboard.
- Turns off the error light.

Record Release Switch Routine

Entry Point: AAO010

Chart: None

Function: Sets an internal switch to reflect the current status of the Record Release switch.

Record Erase Switch Routine

Entry Point: AAP010

Chart: None

Function:

- Reset error counter to zero.
- Clear print work area (PRT2) to blanks.
- Restore functions of Data Entry Keyboard to operational status and turn off error light.
- Reset counters.
- Reset column indicator to 01.
- Reset all right adjust switches to zero.
- Move original card image from correct area (CORRT) to assembly area (ASSM).

Read Key Routine

Entry Point: AAQ010

Chart: None

Function:

- Reads the card to be verified from the secondary hopper into the assembly area (ASSM).
- Moves data from assembly area into Correct area (CORRT).

Program 1 Key Routine

Entry Point: AAR010

Chart: HG

Function:

- Moves the data from the program 1 area (PGM1) to the current control area (PROG).
- Turns on the program 1 bit in the LITE data area.

Program 2 Key Routine

Entry Point: AAS010

Chart: HH

Function:

- Moves the data from the program 2 area (PGM2) to the current control area (PROG).
- Turns on the program 2 bit in the LITE data area.

Program Load Switch Routine

Entry Point: AAT010

Chart: HI

Function:

- Reads a card into the current program area (PROG).
- Checks for an end-of-job card (EOJ).
- Calls EOJ transient routine when encountering an EOJ card.
- Moves the data from the current program area into the specified program control buffer (PGM1 or PGM2).

End or Beginning of Program Defined Field Routine

Entry Point:

- AAU010 – End of field check
- AAU020 – Beginning of field check

Chart: None

Function:

- Checks for end of field by determining if one of the following is satisfied:
  1. Current column of current control area (PROG) is at end of the record.
  2. Current column of current control area (PROG) contains an end-of-field code.
- Checks for beginning of field by determining if one of the following is satisfied:
  1. Current column of current control area is at beginning of the record.
  2. Preceding column of current control area (PROG) contains an end-of-field code.

Adjust Routine

Entry Point: AAV010

Chart: HJ (parts 1 and 2)

Function:

- This routine adjusts the counters to reflect the action taken before this routine was invoked.
- Updates the binary column indicator (BCNT) and decimal column indicator (DCNT) to reflect the next column to be worked on.
- Checks for auto-functions and branches to the appropriate routine.
- Displays the column indicator (LITE) data area.
- When in a right adjust field, the column indicator is advanced to the first non-blank column.

Test for Auto Skip Field Routine

Entry Point: AAW010

Chart: None

Function: Checks the current column of the current control area (PROG) for any one of the codes which indicates an auto skip field.

IOS Interface Routine

Entry Point: DISKNV

Chart: HK

Function: Initiates MFCU IOS operations by translating an IOCS parameter list and passing it to the Full Function MFCU IOS routine.

Exits:

- Normal – To the routine in main program which called it via the address recall register.
- Error -- To Halt/Syslog routine.

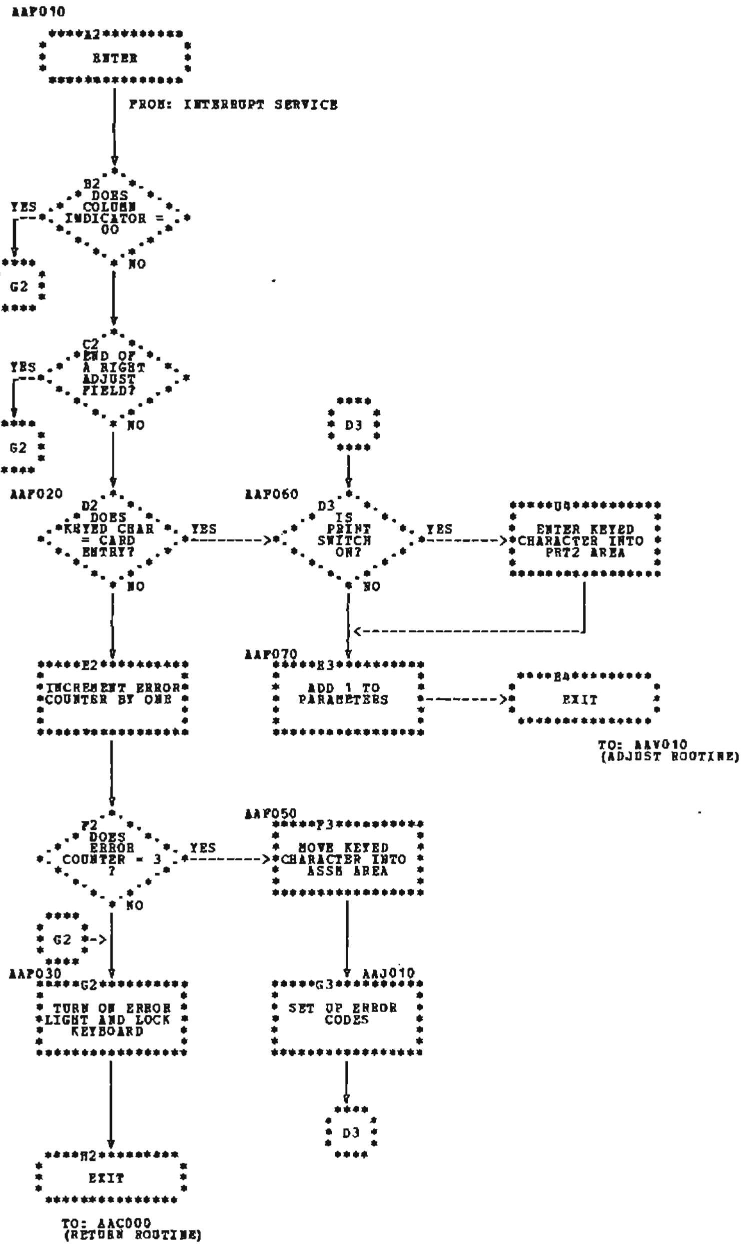

AAF010
A2
ENTER
FROM: INTERRUPT SERVICE
B2
DOES COLUMN INDICATOR = 00
YES
G2
NO
C2
END OF A RIGHT ADJUST FIELD?
YES
G2
NO
D3
AAF020
D2
DOES KEYED CHAR = CARD ENTRY?
YES
NO
AAF060
D3
IS PRINT SWITCH ON?
YES
NO
D4
ENTER KEYED CHARACTER INTO PRT2 AREA
E2
INCREMENT ERROR COUNTER BY ONE
AAF070
E3
ADD 1 TO PARAMETERS
E4
EXIT
TO: AAV010 (ADJUST ROUTINE)
F2
DOES ERROR COUNTER = 3 ?
YES
NO
AAF050
F3
MOVE KEYED CHARACTER INTO ASSM AREA
G2
AAF030
G2
TURN ON ERROR LIGHT AND LOCK KEYBOARD
AAJ010
G3
SET UP ERROR CODES
D3
H2
EXIT
TO: AAC000 (RETURN ROUTINE)

Chart HA. Data Key Routine

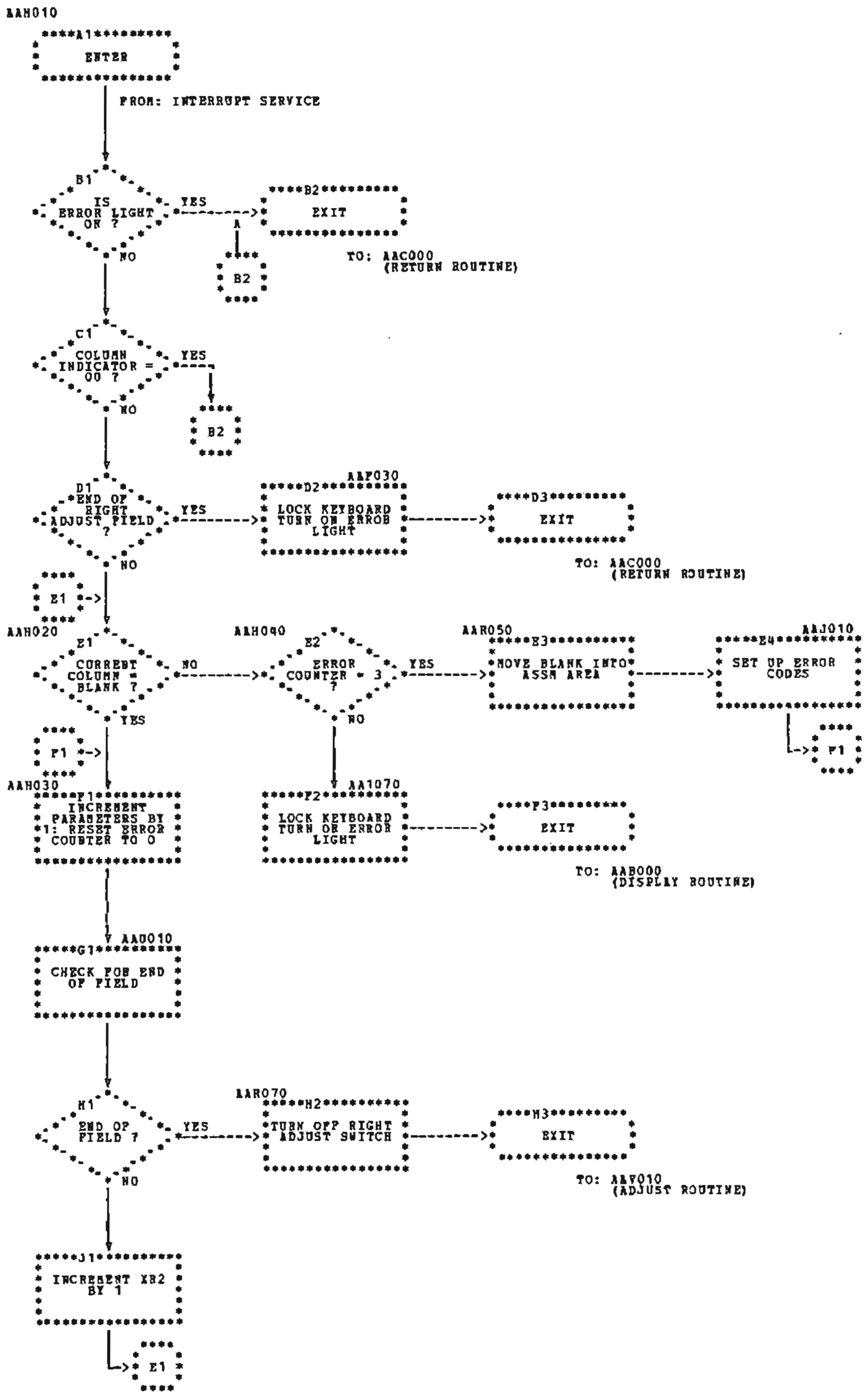


Chart HB. Skip Key Routine

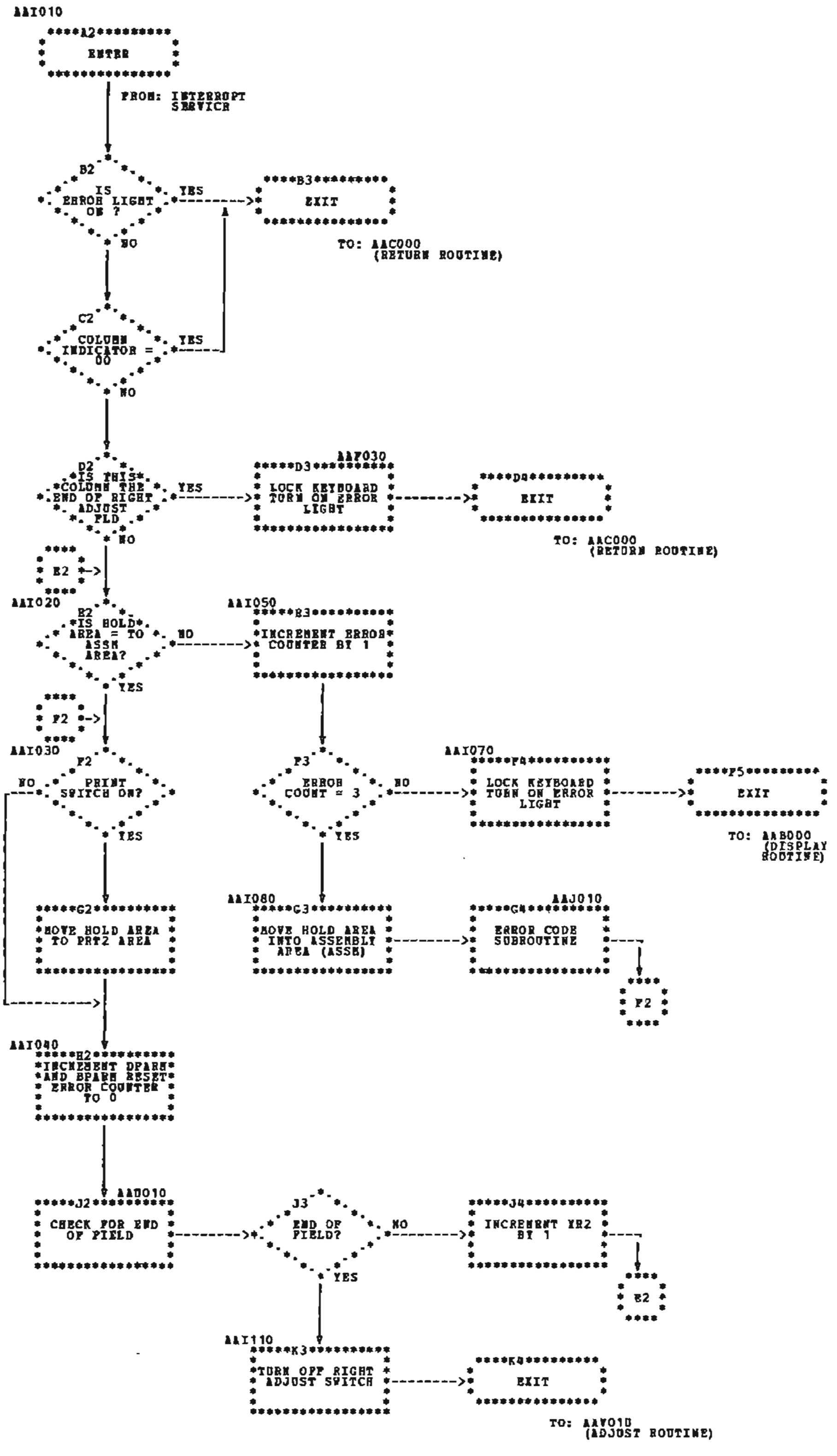

AAI010
A2
ENTER
FROM: INTERRUPT SERVICE
B2
IS ERROR LIGHT ON ?
YES
NO
B3
EXIT
TO: AAC000 (RETURN ROUTINE)
C2
COLUMN INDICATOR = 00
YES
NO
D2
IS THIS COLUMN THE END OF RIGHT ADJUST FLD
YES
NO
AAF030
D3
LOCK KEYBOARD TURN ON ERROR LIGHT
D4
EXIT
TO: AAC000 (RETURN ROUTINE)
E2
AAI020
E2
IS HOLD AREA = TO ASSM AREA?
NO
YES
AAI050
E3
INCREMENT ERROR COUNTER BY 1
F2
AAI030
F2
PRINT SWITCH ON?
NO
YES
F3
ERROR COUNT = 3
NO
YES
AAI070
F4
LOCK KEYBOARD TURN ON ERROR LIGHT
F5
EXIT
TO: AAB000 (DISPLAY ROUTINE)
G2
MOVE HOLD AREA TO PRT2 AREA
AAI080
G3
MOVE HOLD AREA INTO ASSEMBLY AREA (ASSM)
AAJ010
G4
ERROR CODE SUBROUTINE
F2
AAI040
H2
INCREMENT DPARM AND DPARM RESET ERROR COUNTER TO 0
AAD010
J2
CHECK FOR END OF FIELD
J3
END OF FIELD?
NO
YES
J4
INCREMENT XR2 BY 1
E2
AAI110
K3
TURN OFF RIGHT ADJUST SWITCH
K4
EXIT
TO: AAY010 (ADJUST ROUTINE)

Chart HC. Dup Key Routine

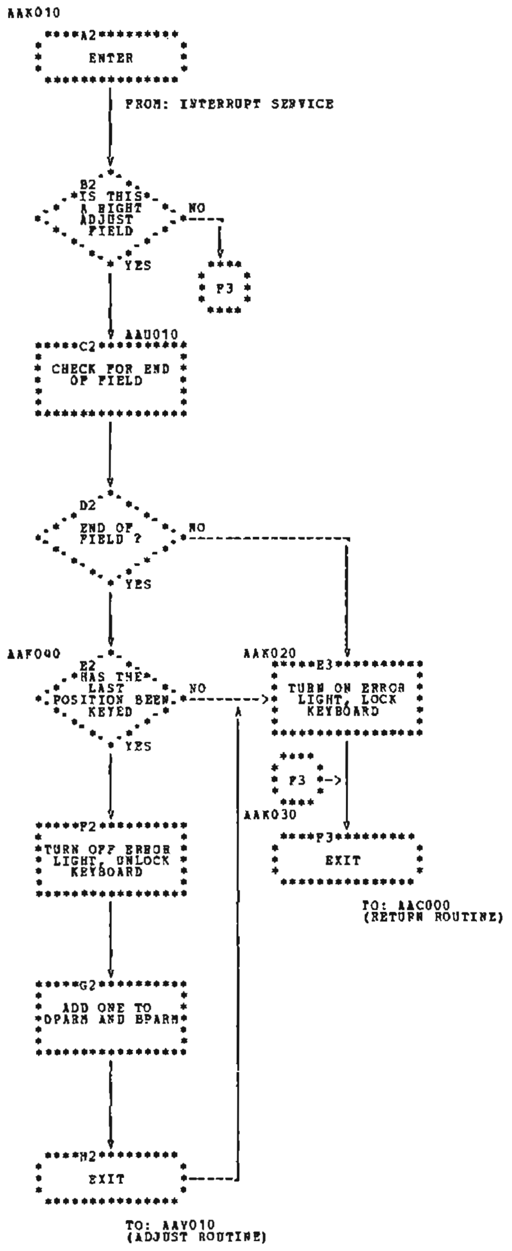

AAK010
A2
ENTER
FROM: INTERRUPT SERVICE
B2
IS THIS A RIGHT ADJUST FIELD
NO
YES
F3
AAU010
C2
CHECK FOR END OF FIELD
D2
END OF FIELD ?
NO
YES
AAF040
E2
HAS THE LAST POSITION BEEN KEYED
NO
YES
AAK020
E3
TURN ON ERROR LIGHT, LOCK KEYBOARD
F3
AAK030
F3
EXIT
TO: AAC000 (RETURN ROUTINE)
F2
TURN OFF ERROR LIGHT, UNLOCK KEYBOARD
G2
ADD ONE TO DPARM AND BPARM
H2
EXIT
TO: AAV010 (ADJUST ROUTINE)

Chart HD. Right Adjust Key Routine

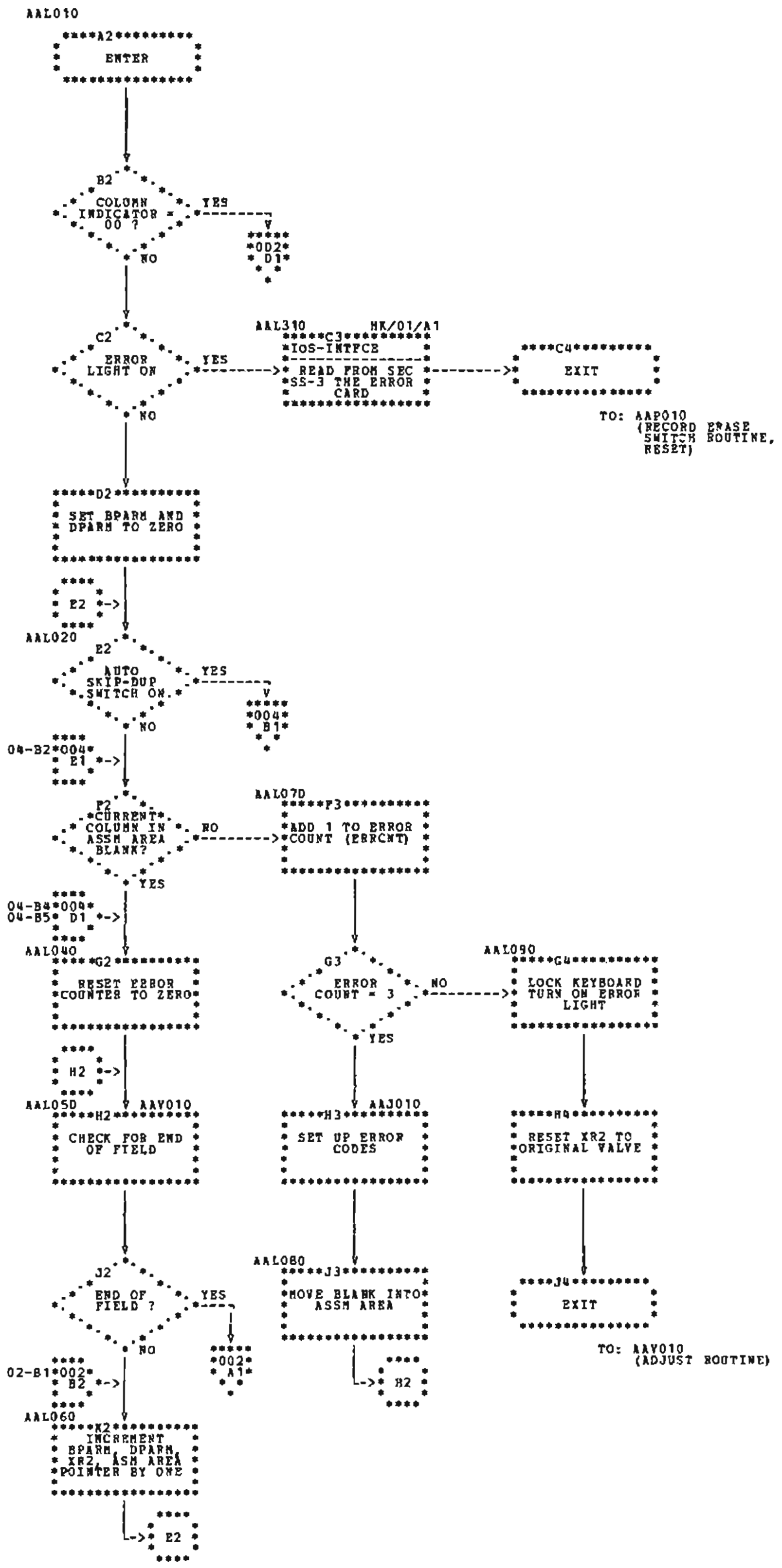

AAL010
A2
ENTER
B2
COLUMN INDICATOR = 00 ?
YES
002 D1
NO
C2
ERROR LIGHT ON
YES
AAL310
HK/01/A1
C3
IOS-INTFCE
READ FROM SEC SS-3 THE ERROR CARD
C4
EXIT
TO: AAP010 (RECORD ERASE SWITCH ROUTINE, RESET)
NO
D2
SET BPARM AND DPARM TO ZERO
E2
AAL020
E2
AUTO SKIP-DUP SWITCH ON.
YES
004 B1
NO
04-B2
004 E1
F2
CURRENT COLUMN IN ASSM AREA BLANK?
NO
AAL07D
F3
ADD 1 TO ERROR COUNT (ERRCNT)
YES
04-B4
04-B5
004 D1
AAL040
G2
RESET ERROR COUNTER TO ZERO
G3
ERROR COUNT = 3
NO
AAL090
G4
LOCK KEYBOARD TURN ON ERROR LIGHT
YES
H2
AAL05D
AAV010
H2
CHECK FOR END OF FIELD
AAJ010
H3
SET UP ERROR CODES
H4
RESET XR2 TO ORIGINAL VALVE
J2
END OF FIELD ?
YES
002 A1
NO
AAL080
J3
MOVE BLANK INTO ASSM AREA
H2
J4
EXIT
TO: AAV010 (ADJUST ROUTINE)
02-B1
002 B2
AAL060
K2
INCREMENT BPARM, DPARM, XR2, ASM AREA POINTER BY ONE
E2

Chart HE. Release Key Routine (Part 1 of 4)

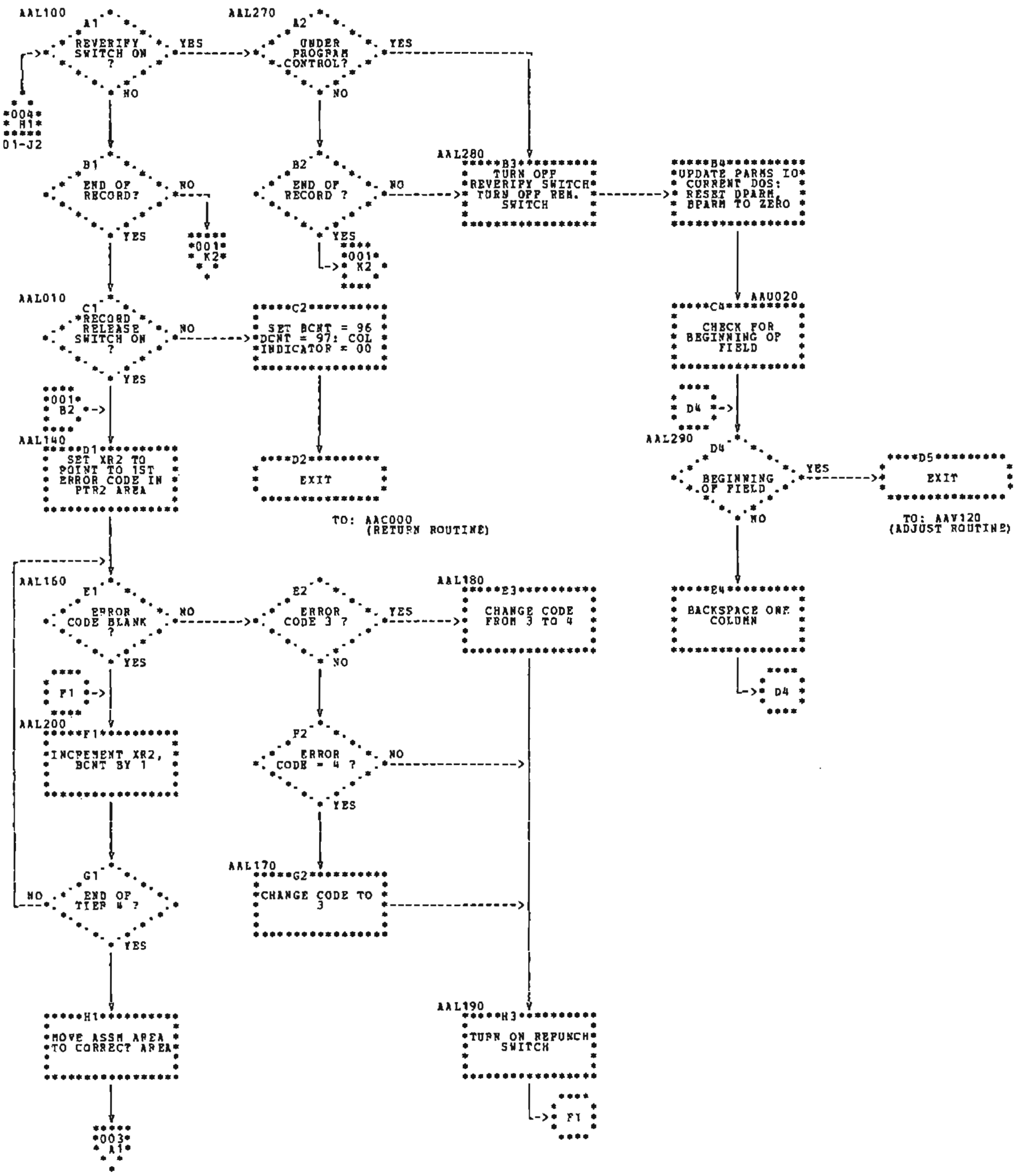


Chart HE. Release Key Routine (Part 2 of 4)

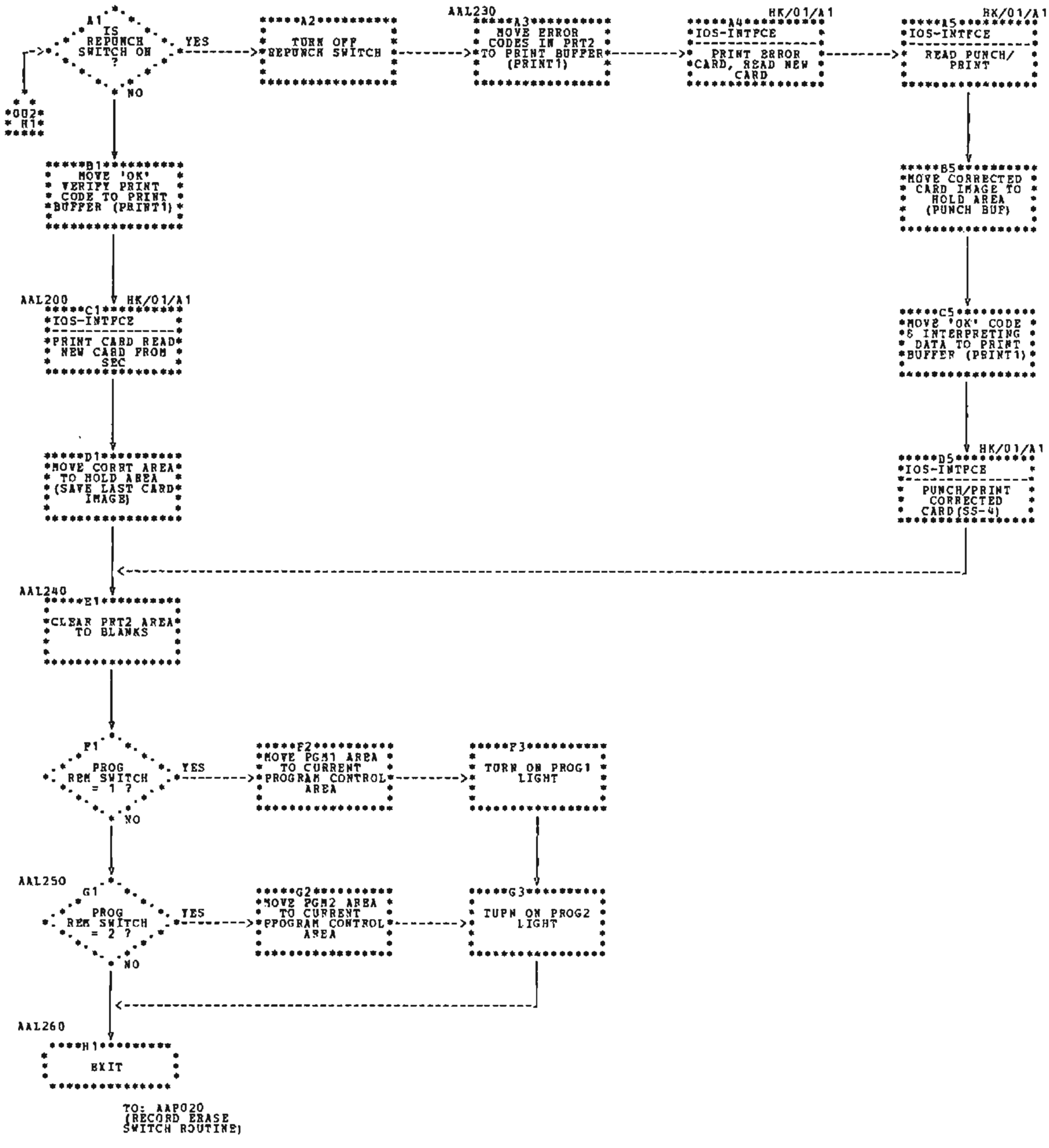

A1
IS REPUNCH SWITCH ON ?
YES
NO
002 H1
A2
TURN OFF REPUNCH SWITCH
AAL230
A3
MOVE ERROR CODES IN PRT2 TO PRINT BUFFER (PRINT1)
HK/01/A1
A4
IOS-INTFCE
PRINT ERROR CARD, READ NEW CARD
HK/01/A1
A5
IOS-INTFCE
READ PUNCH/ PRINT
B1
MOVE 'OK' VERIFY PRINT CODE TO PRINT BUFFER (PRINT1)
B5
MOVE CORRECTED CARD IMAGE TO HOLD AREA (PUNCH BUF)
AAL200
HK/01/A1
C1
IOS-INTFCE
PRINT CARD READ NEW CARD FROM SEC
C5
MOVE 'OK' CODE & INTERPRETING DATA TO PRINT BUFFER (PRINT1)
D1
MOVE CORRT AREA TO HOLD AREA (SAVE LAST CARD IMAGE)
HK/01/A1
D5
IOS-INTFCE
PUNCH/PRINT CORRECTED CARD(SS-4)
AAL240
E1
CLEAR PRT2 AREA TO BLANKS
F1
PROG REM SWITCH = 1 ?
YES
NO
F2
MOVE PGM1 AREA TO CURRENT PROGRAM CONTROL AREA
F3
TURN ON PROG1 LIGHT
AAL250
G1
PROG REM SWITCH = 2 ?
YES
NO
G2
MOVE PGM2 AREA TO CURRENT PROGRAM CONTROL AREA
G3
TURN ON PROG2 LIGHT
AAL260
H1
EXIT
TO: AAP020 (RECORD ERASE SWITCH ROUTINE)

Chart HE. Release Key Routine (Part 3 of 4)

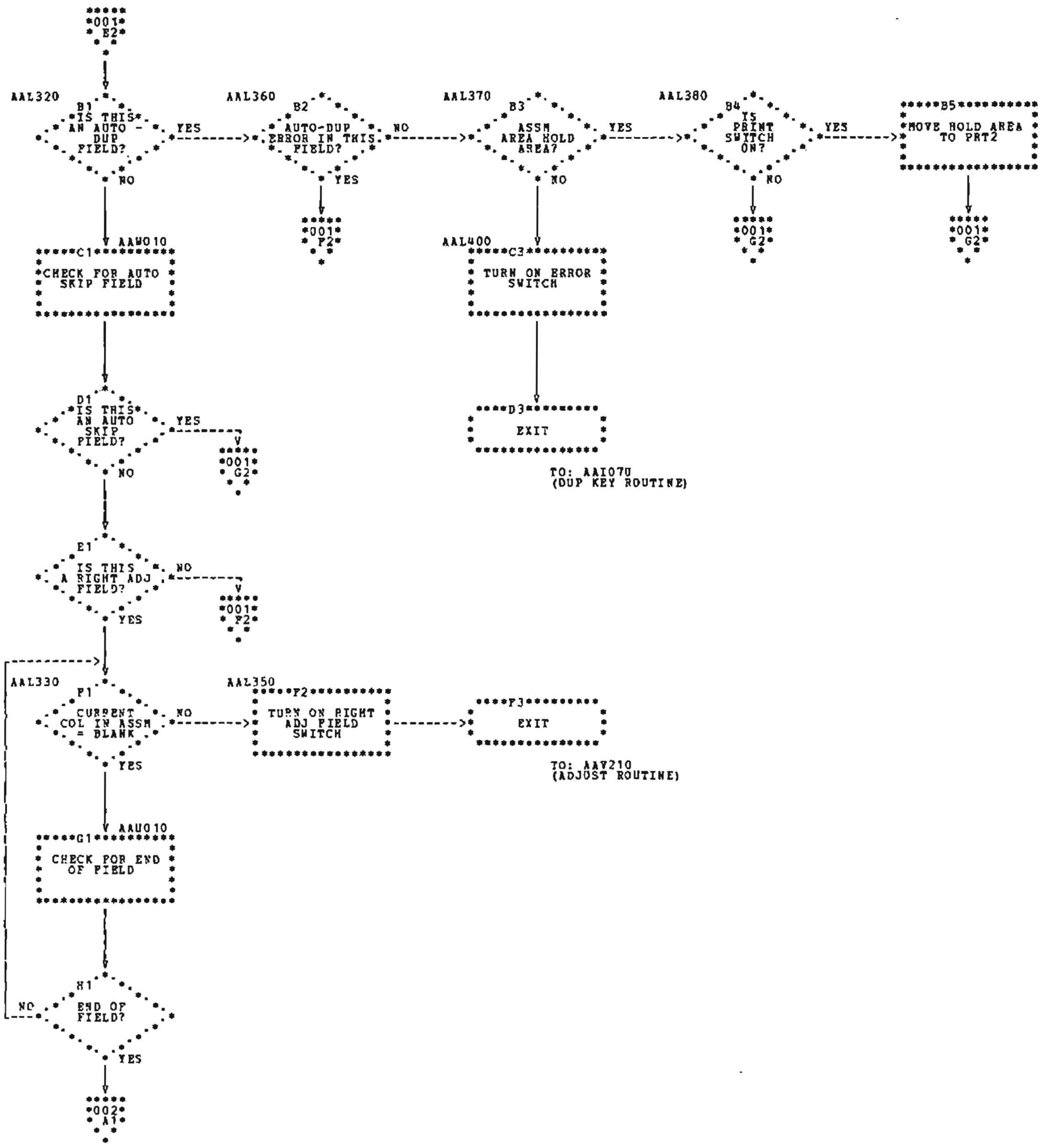


Chart HE. Release Key Routine (Part 4 of 4)

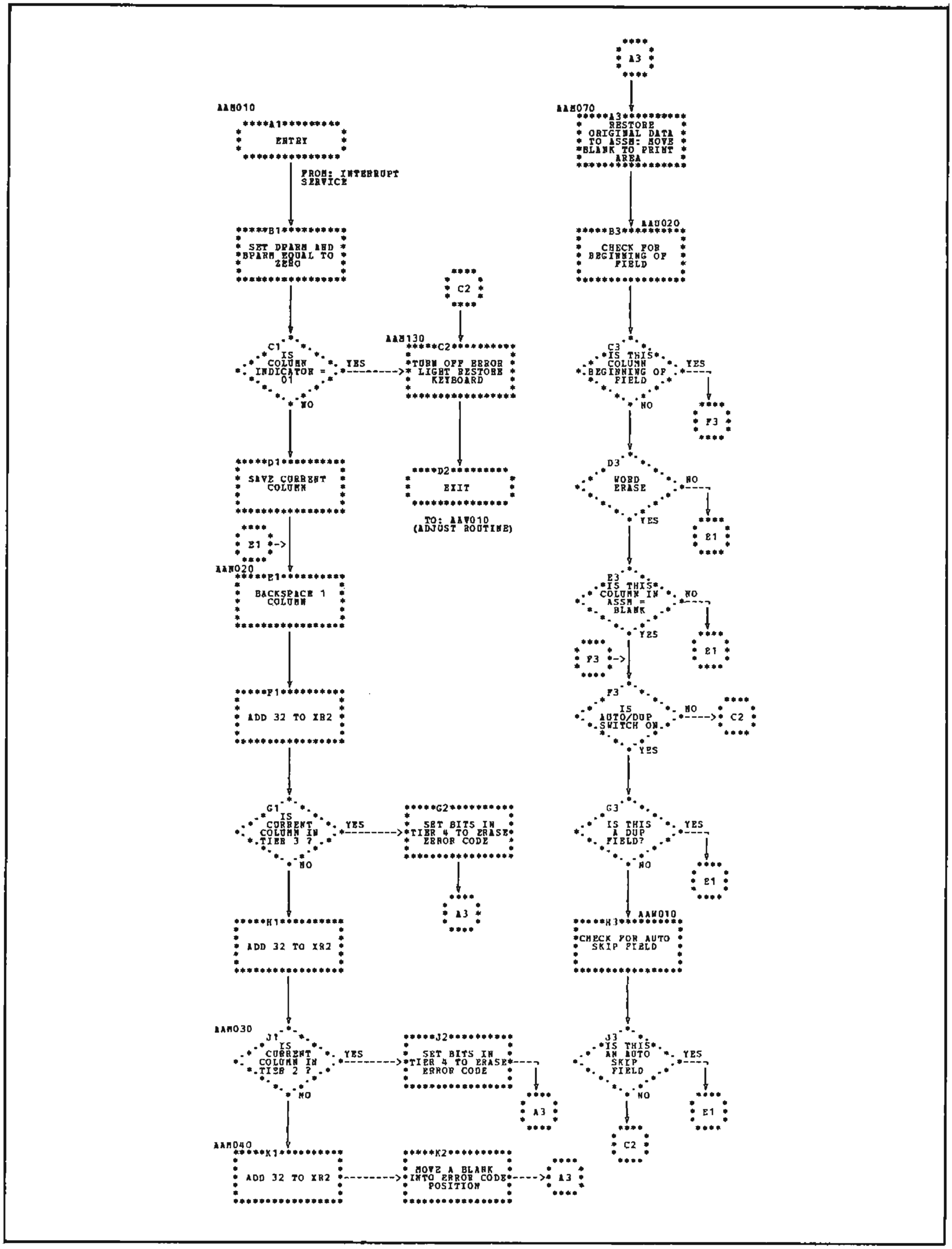

AAM010
A1 ENTRY
FROM: INTERRUPT SERVICE
B1 SET DPARM AND BPARM EQUAL TO ZERO
C1 IS COLUMN INDICATOR = 01
YES
NO
C2
AAM130
C2 TURN OFF ERROR LIGHT RESTORE KEYBOARD
D1 SAVE CURRENT COLUMN
D2 EXIT
TO: AAV010 (ADJUST ROUTINE)
E1
AAM020
E1 BACKSPACE 1 COLUMN
F1 ADD 32 TO XR2
G1 IS CURRENT COLUMN IN TIER 3 ?
YES
NO
G2 SET BITS IN TIER 4 TO ERASE ERROR CODE
A3
H1 ADD 32 TO XR2
AAM030
J1 IS CURRENT COLUMN IN TIER 2 ?
YES
NO
J2 SET BITS IN TIER 4 TO ERASE ERROR CODE
A3
AAM040
K1 ADD 32 TO XR2
K2 MOVE A BLANK INTO ERROR CODE POSITION
A3
A3
AAM070
A3 RESTORE ORIGINAL DATA TO ASSM: MOVE BLANK TO PRINT AREA
AAU020
B3 CHECK FOR BEGINNING OF FIELD
C3 IS THIS COLUMN BEGINNING OF FIELD
YES
F3
NO
D3 WORD ERASE
NO
E1
YES
E3 IS THIS COLUMN IN ASSM = BLANK
NO
E1
YES
F3
F3 IS AUTO/DUP SWITCH ON
NO
C2
YES
G3 IS THIS A DUP FIELD?
YES
E1
NO
AAW010
H3 CHECK FOR AUTO SKIP FIELD
J3 IS THIS AN AUTO SKIP FIELD
YES
E1
NO
C2

Chart HF. Field Erase Key Routine

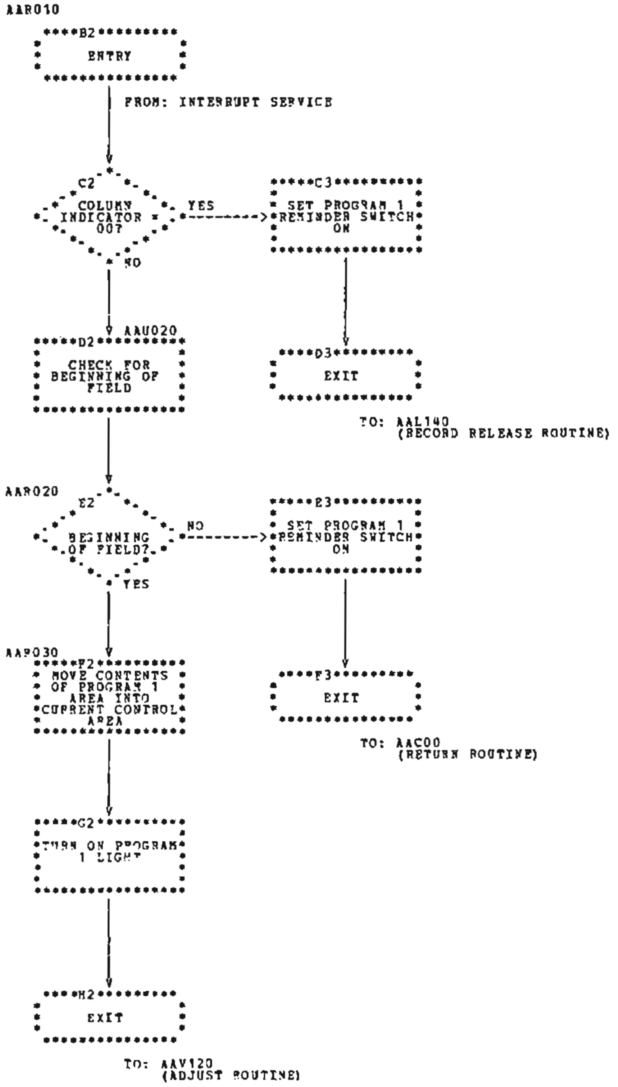

AAR010
B2
ENTRY
FROM: INTERRUPT SERVICE
C2
COLUMN INDICATOR = 00?
YES
NO
C3
SET PROGRAM 1 REMINDER SWITCH ON
AAU020
D2
CHECK FOR BEGINNING OF FIELD
D3
EXIT
TO: AAL140 (RECORD RELEASE ROUTINE)
AAR020
E2
BEGINNING OF FIELD?
NO
YES
E3
SET PROGRAM 1 REMINDER SWITCH ON
AAR030
F2
MOVE CONTENTS OF PROGRAM 1 AREA INTO CURRENT CONTROL AREA
F3
EXIT
TO: AAC00 (RETURN ROUTINE)
G2
TURN ON PROGRAM 1 LIGHT
H2
EXIT
TO: AAV120 (ADJUST ROUTINE)

Chart HG. Program 1 Key Routine

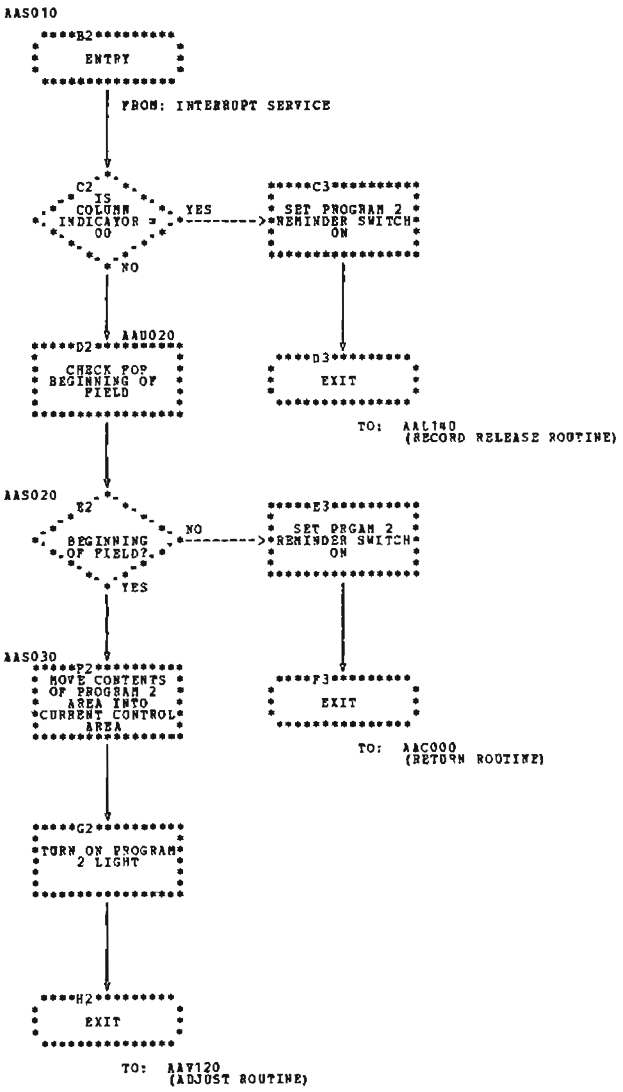

AAS010
B2
ENTRY
FROM: INTERRUPT SERVICE
C2
IS COLUMN INDICATOR = 00
YES
NO
C3
SET PROGRAM 2 REMINDER SWITCH ON
D2
AAU020
CHECK FOR BEGINNING OF FIELD
D3
EXIT
TO: AAL140 (RECORD RELEASE ROUTINE)
AAS020
E2
BEGINNING OF FIELD?
NO
YES
E3
SET PRGAM 2 REMINDER SWITCH ON
AAS030
F2
MOVE CONTENTS OF PROGRAM 2 AREA INTO CURRENT CONTROL AREA
F3
EXIT
TO: AAC000 (RETURN ROUTINE)
G2
TURN ON PROGRAM 2 LIGHT
H2
EXIT
TO: AAV120 (ADJUST ROUTINE)

Chart HH. Program 2 Key Routine

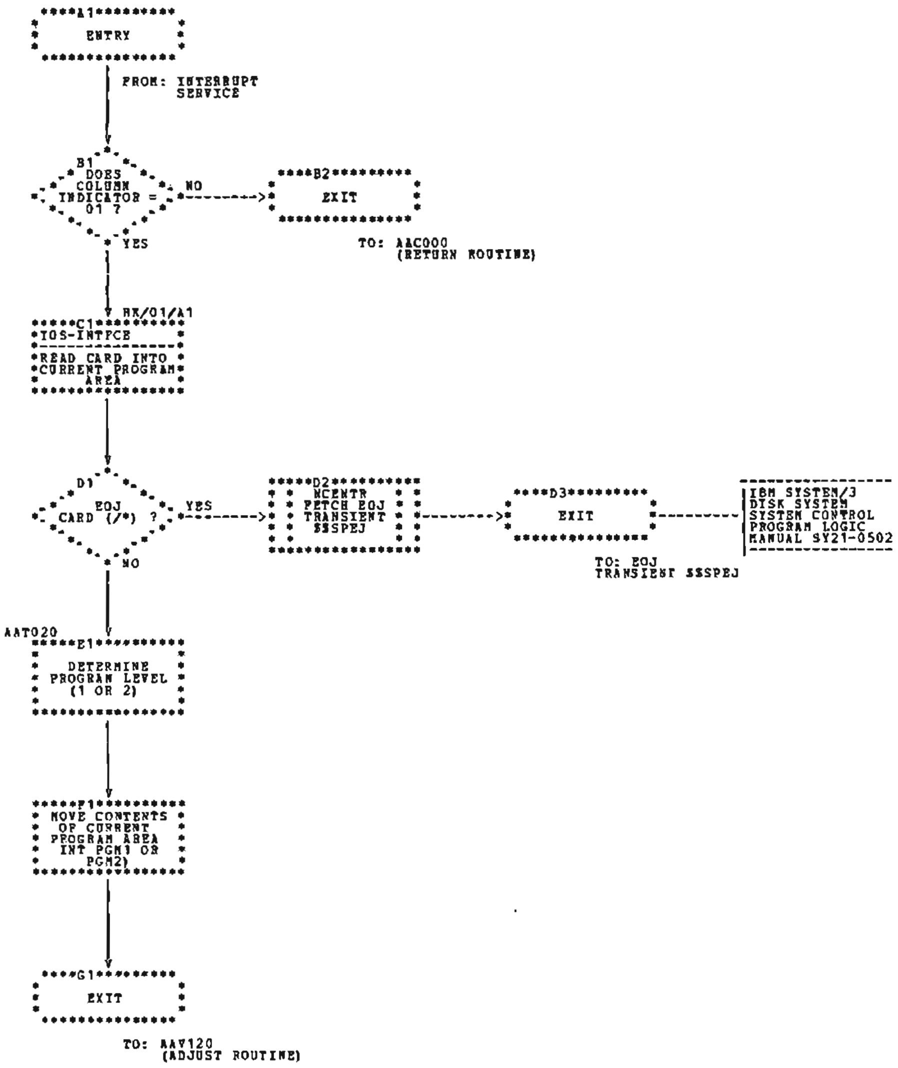

A1
ENTRY
FROM: INTERRUPT SERVICE
B1
DOES COLUMN INDICATOR = 01 ?
NO
B2
EXIT
TO: AAC000 (RETURN ROUTINE)
YES
RK/01/A1
C1
IOS-INTPCE
READ CARD INTO CURRENT PROGRAM AREA
D1
EOJ CARD (/*) ?
YES
D2
NCENTR
FETCH EOJ TRANSIENT $$SPEJ
D3
EXIT
TO: EOJ TRANSIENT $$SPEJ
IBM SYSTEM/3 DISK SYSTEM SYSTEM CONTROL PROGRAM LOGIC MANUAL SY21-0502
NO
AAT020
E1
DETERMINE PROGRAM LEVEL (1 OR 2)
F1
MOVE CONTENTS OF CURRENT PROGRAM AREA INT PGM1 OR PGM2)
G1
EXIT
TO: AAV120 (ADJUST ROUTINE)

Chart HI. Program Load Routine

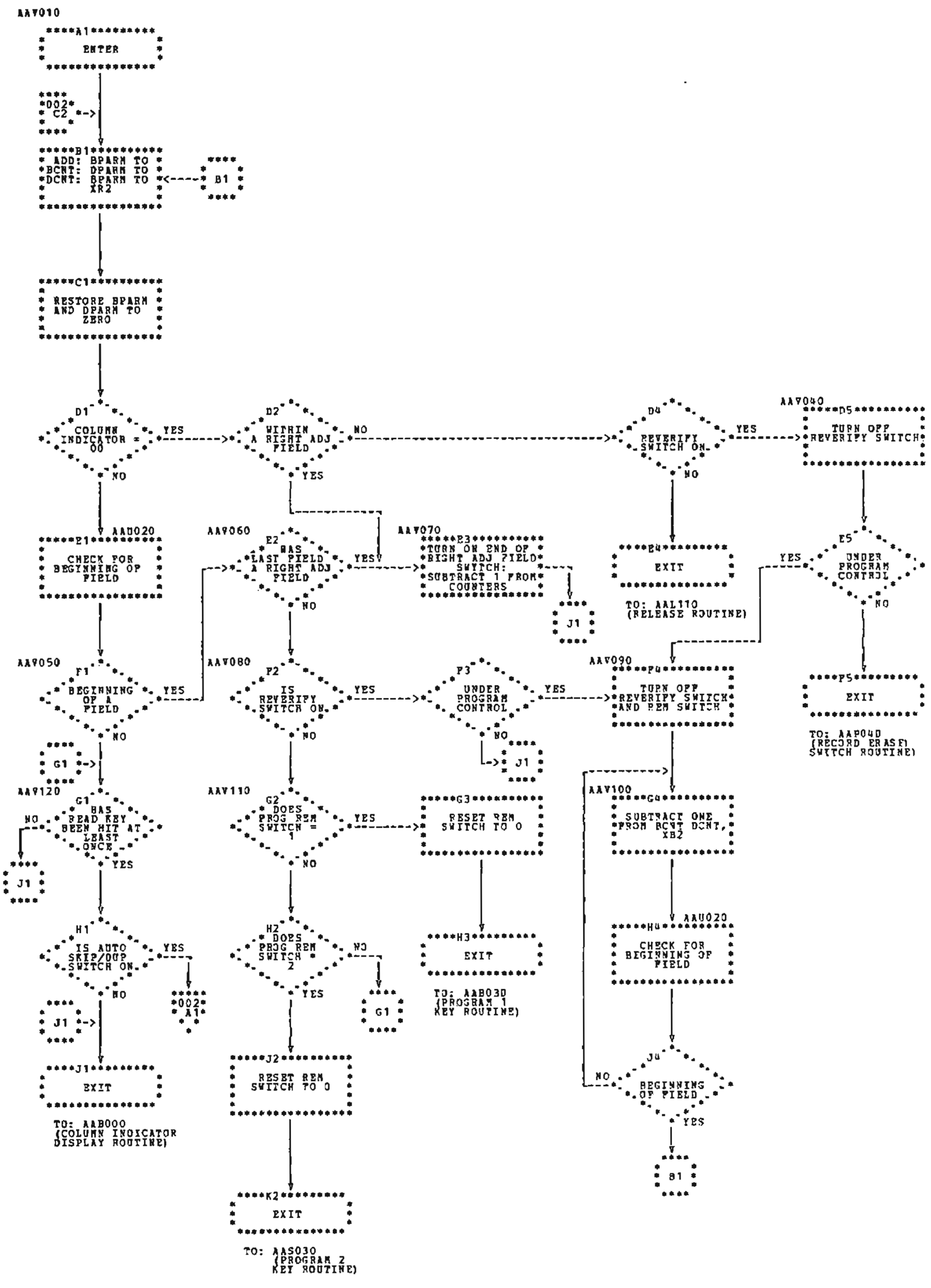


Chart HJ. Adjust Routine (Part 1 of 2)

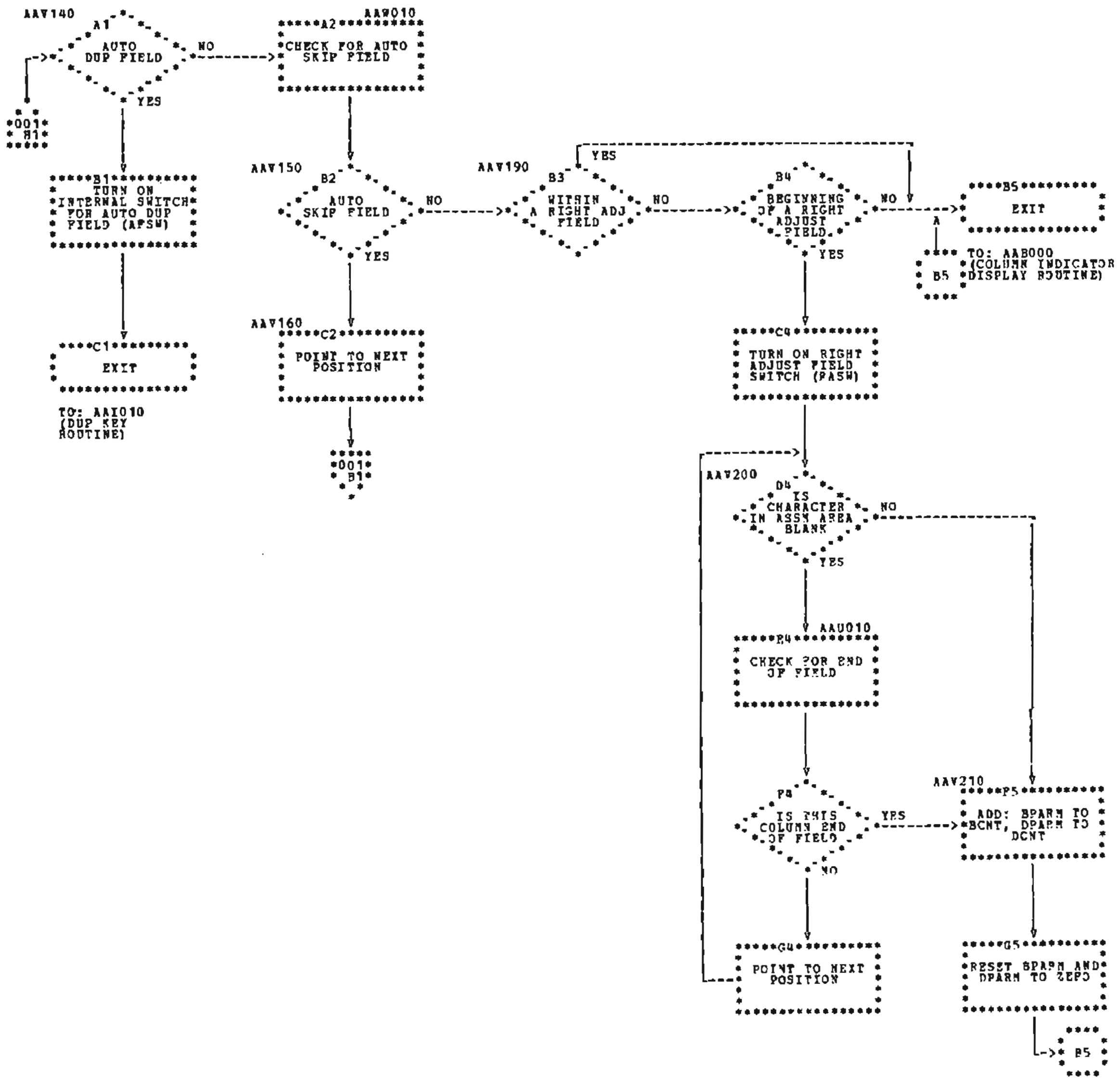

AAV140
A1
AUTO DUP FIELD
NO
YES
001
B1
AAW010
A2
CHECK FOR AUTO SKIP FIELD
B1
TURN ON INTERNAL SWITCH FOR AUTO DUP FIELD (AFSW)
C1
EXIT
TO: AAI010 (DUP KEY ROUTINE)
AAV150
B2
AUTO SKIP FIELD
NO
YES
AAV160
C2
POINT TO NEXT POSITION
001
B1
AAV190
B3
WITHIN A RIGHT ADJ FIELD
YES
NO
B4
BEGINNING OF A RIGHT ADJUST FIELD
NO
YES
B5
EXIT
A
B5
TO: AAB000 (COLUMN INDICATOR DISPLAY ROUTINE)
C4
TURN ON RIGHT ADJUST FIELD SWITCH (RASW)
AAV200
D4
IS CHARACTER IN ASSM AREA BLANK
NO
YES
AAU010
E4
CHECK FOR END OF FIELD
F4
IS THIS COLUMN END OF FIELD
YES
NO
AAV210
F5
ADD: BPARM TO BCNT, DPARM TO DCNT
G4
POINT TO NEXT POSITION
G5
RESET BPARM AND DPARM TO ZERO
B5

Chart HJ. Adjust Routine (Part 2 of 2)

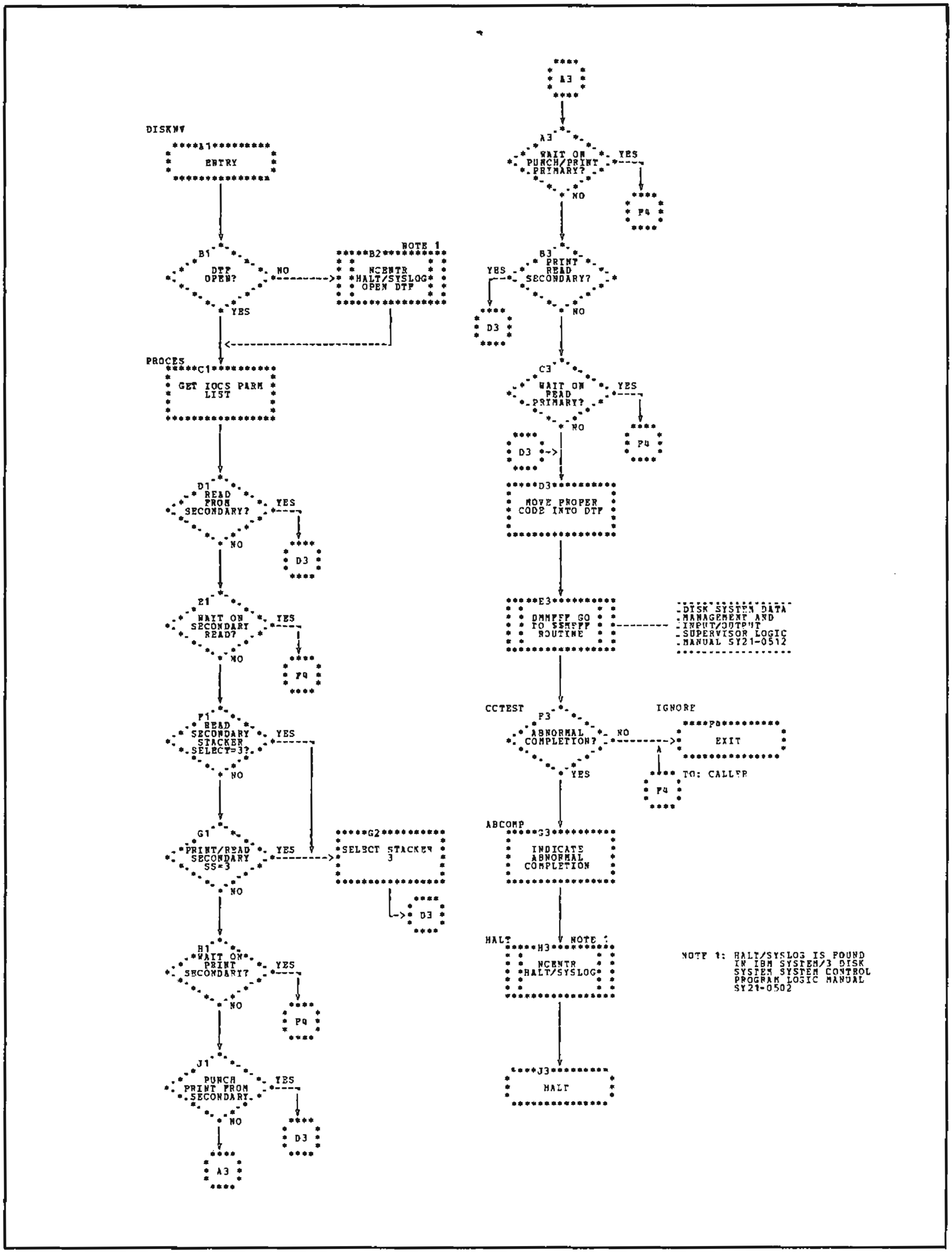


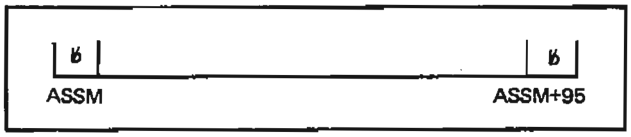
Chart HK. IOS Interface Routine

Section 4. Data Area Formats

This section describes data areas that are used by more than two routines.

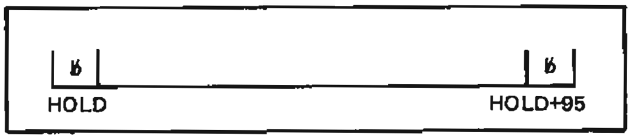
Assembly Area – ASSM

This 96-byte area is initially filled with blanks. It is used as a read buffer for both hoppers and as a work area for the verified card image. This area is aligned on a hexadecimal 80 boundary.



Hold Area -- HOLD

This 96-byte area is initially filled with blanks. After the first card is verified, this area is used to hold the image of the last card released. The hold area is also used as the punch buffer. This area is aligned on a hexadecimal 80 boundary.



Correct Area – CORRT

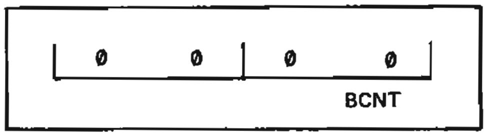
This 96-byte area is initially filled with blanks and is used to hold the image of the card being verified.

ƀ
ƀ
CORRT
CORRT+95

Binary Column Indicator -- BCNT

This 2-byte area is initially set to zero. This indicator is used to index through the card work areas on a column by column basis. The setting of this indicator reflects the number of card columns that have been verified. The possible range is from 0 to 96 (decimal values).

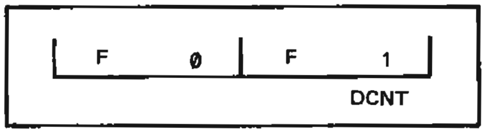
Note: Data is in a binary format.



Decimal Column Indicator – DCNT

This 2-byte area is initially set to 01 (decimal value). This indicator is used to reflect the column that is currently to be operated on. The value of DCNT is used in selecting the respective light combinations displayed on the column indicator on the front of the Data Entry Keyboard.

Note: Data is in a zoned decimal format.



Column Indicator -- LITE

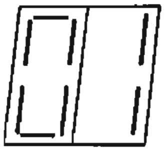
This is a 2-byte area, initially set to X'EE', X'24' (see Figure 6-4). This area is used as an input area by the LIO instruction that displays the column indicator. This indicator is initially displayed at 01.

Stick-Light Table – TAB

This is a 10-byte area consisting of hexadecimal values for the decimal numbers 0-9. The decimal column indicator (DCNT) is used in referencing this table (see Figure 6-5). The entries from this table (TAB) are placed in LITE.

EE	24
Lite	Lite +1
2nd Byte	1st Byte

Displayed As:



	Bit 0	Bit 1	Bit 2	Bit 3	Bit 4	Bit 5	Bit 6	Bit 7
1st Byte (Unit's)	Lighted Segment E	Lighted Segment D	Lighted Segment F	Lighted Segment C	Lighted Segment B	Lighted Segment G	Lighted Segment A	Prog 2 Indicator
2nd Byte (Ten's)	Lighted Segment E	Lighted Segment D	Lighted Segment F	Lighted Segment C	Lighted Segment B	Lighted Segment G	Lighted Segment A	Prog 1 Indicator

Hex Code	Digit	Lighted Segments
EE	0	ABDEFG
24	1	FG
BA	2	ABCEF
B6	3	ACEFG
74	4	CDFG
D6	5	ACDEG
DE	6	ABCDEG
A4	7	EFG
FE	8	ABCDEFG
F6	9	ACDEFG

Keyboard Column Indicator

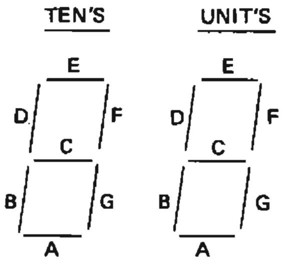


Figure 6-4. Column Indicator Data Area and Display Values

X'EE'	X'24'	X'BA'	X'B6'	X'74'	X'D6'	X'DE'	X'A4'	X'FE'	X'F6'
TAB									TAB+9

Hex Byte Value	Displayed on Keyboard as:
X'EE'	0
X'24'	1
X'BA'	2
X'B6'	3
X'74'	4
X'D6'	5
X'DE'	6
X'A4'	7
X'FE'	8
X'F6'	9

Figure 6-5. Stick Light Table and Display Values

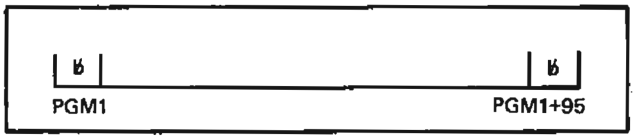
Current Control Area – PROG

This is a 96-byte area that is initially set in every byte position to X'D4' (code for end of field and lower shift). When the program is under manual control, all bytes of this area are set to X'D4'. This area always contains the image of the program card that is in current control.

X'D4'
X'D4'
PROG
PROG+95

Program 1 – PGM1

This is a 96-byte storage area used to contain the image of the program 1 control card. This area is initially filled with blanks.



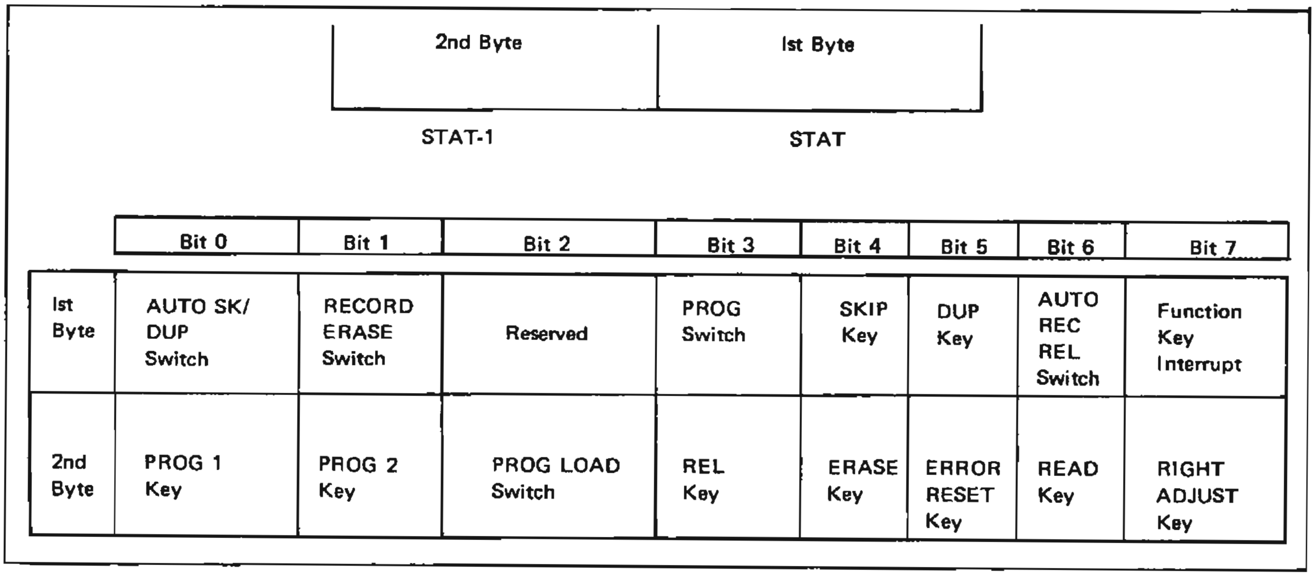
Program 2 – PGM2

This is a 96-byte storage area used to contain the image of the program 2 control card. This area is initially filled with blanks.

b̸
b̸
PGM2
PGM2+95

Sense Table – STAT/DATA

This is a 4-byte area that is used in detecting the source of an interrupt. The first two bytes are referenced by the label STAT and indicate whether a function key was the cause of the interrupt (see Figure 6-6). The second two bytes are referenced by the label DATA and indicate whether a data key was the source of the interrupt (see Figure 6-7). The sense table receives the sense data from the SNS instructions to the Data Entry Keyboard.



	Bit 0	Bit 1	Bit 2	Bit 3	Bit 4	Bit 5	Bit 6	Bit 7
Ist Byte	AUTO SK/ DUP Switch	RECORD ERASE Switch	Reserved	PROG Switch	SKIP Key	DUP Key	AUTO REC REL Switch	Function Key Interrupt
2nd Byte	PROG 1 Key	PROG 2 Key	PROG LOAD Switch	REL Key	ERASE Key	ERROR RESET Key	READ Key	RIGHT ADJUST Key

Figure 6-6. Function Key Sense Table

2nd Byte	Ist Byte
DATA-1	DATA

	Bit 0	Bit 1	Bit 2	Bit 3	Bit 4	Bit 5	Bit 6	Bit 7
Ist Byte	PRINT Switch	Reserved	LOWER SHIFT Key	Invalid Character*	Reserved	MULT PCH Key	Reserved	Data Key Interrupt
2nd Byte	Data Character Keyed (EBCDIC)							

* The presence of this bit indicates that numeric shift was programmed and a character other than 0 through 9 or space was keyed.

NOTE: If the bit = 1, that key or switch is on.
If the bit = 0, that key or switch is off.

Figure 6-7. Data Key Sense Table

Control Code -- CCODE

This is a 1-byte area used as the Q code for the SIO instruction in the Return routine. This area is initially set to hexadecimal 0F (see Figure 6-8).

X'OF'

CCODE

Control Code Bits

Bit 0–Programmed Numeric Mode
Bit 1–Programmed Lower Shift
Bit 2–Error Indicator
Bit 3–Spare
Bit 4–Restore Data Key
Bit 5–Unlock Data Key
Bit 6–Enable/Disable Interrupt
Bit 7–Reset Interrupt

Note: If bit = 1, that function is enabled.
If bit = 0, that function is disabled.

Figure 6-8. Control Code Data Area

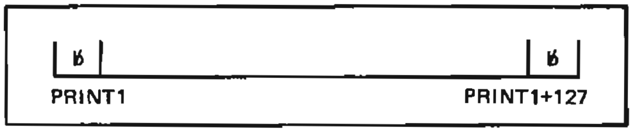
Remember Switch -- REM

This is a 1-byte area initially set to hexidecimal 00. This area is used to indicate a change in the current control area (PROG).

Switch Setting	*Meaning*
0	Do not change current control area (PROG).
1	Move program 1 (PGM1) contents into current control area (PROG).
2	Move program 2 (PGM2) contents into current control area (PROG).

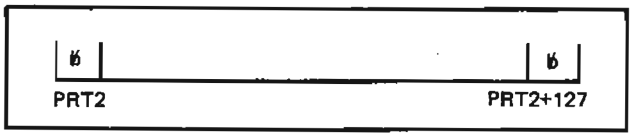
Print Area -- PRINT1

This 128-byte area is initially filled with blanks. It is used as the print buffer and is aligned on a hexidecimal 80 boundary.



Print 2 Area -- PRT2

This 128-byte area is initially filled with blanks. The PRT2 area is used as a work area for printing. The first 96 bytes are used for interpreting the corrected card. The last 32 bytes are used for building the error codes that are printed on the fourth tier of the incorrect card (see Figure 6-9 for codes).



Copyright

This 46-byte area contains the program number for this program and copyright information as follows: 5702-UT1ƀCOPYRIGHTƀIBMƀCORPƀ1970. (The ƀ represents a blank.) The remainder of the area is filled with blanks.

MFCU IOCS Parameters -- IOB

Each parameter is 6 bytes long. They contain the IOCS operation codes. The codes are passed to the IOS Interface routine which places an equivalent code in the DTF. The codes are shown in Figure 6-10.

Buffer-Associated IOB -- PUIOB, RDIOB

See *Part 4. Data Recording, Section 4. Data Area Formats, Buffer-Associated IOB – PUIOB, RDIOB* for definition of these areas.

Define the File -- DTF

See *Part 4. Data Recording, Section 4. Data Area Formats, Define the File – DTF.*

Data Area Activity

Figure 6-11 shows which Data Verifying routines use the data area described in this section.

Error Code Printed on Card	Tiers Containing Errors	Hexadecimal Value in PRT2
1	1	X'01'
2	2	X'02'
3	3	X'04'
4	1 and 2	X'03'
5	1 and 3	X'05'
6	2 and 3	X'06'
7	1, 2 and 3	X'07'

Figure 6-9. Error Codes Entered in PRT2 Area

AREA	CODE	DESCRIPTION
RDS	X'810000800000'	Read secondary
WRDS	X'010000800000'	Wait on read secondary
RDS 3	X'810007800000'	Read secondary from stacker select 3
PRRDS 3	X'850007800000'	Print, read secondary from stacker select 3
WPRS	X'040000800000'	Wait on print secondary
PNPRP 4	X'E00004800000'	Punch, print, on primary
WPNPRP	X'600000800000'	Wait on punch, print from primary
PRRDS	X'850000800000'	Print, read from secondary
RDP	X'900000800000'	Read from primary
WRDP	X'100000800000'	Wait on read from primary

Note: The wait codes, WRDS, WPRS, WPNPRP, and WRDP are ignored by the IOS Interface routine because the Full Function MFCU IOS routine ($$MFFF) automatically handles them before performing an operation such as read or print.

Figure 6-10. IOCS Parameter List

Data Areas / Using Routines	ASSM.	HOLD	PRT2	PRINT1	PROG	PGM1	PGM2	DATA/STAT	CCODE	BCNT	DCNT	REM	LITE	TAB	CORRT	IOCS PARAMETER LIST	DTF
Initializing Routine								X								X	
Display Routine											X		X	X			
Return Routine					X				X				X				
Interrupt Handler Routine								X									
Interrupt Service Routine					X			X					X				
Data Key Routine	X		X		X			X	X	X							
Invalid Character Routine									X								
Skip Key Routine	X									X							
Dup Key Routine	X	X	X						X	X							
Right Adjust Key Routine									X								
Release Key Routine	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	
Field Erase Key Routine	X		X		X			X	X	X		X			X		
Error Reset Key Routine									X	X							
Record Release Switch Routine										X							
Record Erase Switch Routine	X		X						X	X	X	X			X		
Read Key Routine	X								X	X					X		
Program 1 Key Routine					X	X			X	X		X	X				
Program 2 Key Routine					X		X		X	X		X	X				
Program Load Switch Routine					X	X	X			X			X				
Adjust Routine	X				X			X	X	X	X	X					
Error Code Routine			X							X							
IOS Interface Routine																X	X

Figure 6-11. Data Area Activity Chart

Section 1. Introduction

The 80–96 Conversion program is a disk resident program designed to convert the contents of 80–column punched cards to 96–column cards.

The user supplies conversion language specifications which design the 80–96 Conversion program to his particular needs. He can:

- Convert data cards keeping the same format in both.
- Convert and reformat data cards.
- Correct erroneous data.
- Restructure multi-punched coding schemes.
- Change position of the sign of the field.

The user also has the option of interpreting his 96–column cards.

Note: Throughout this chapter, 80–column cards are referred to as source cards; 96–column cards are referred to as destination cards.

System Requirements

The 80–96 Conversion program requires:

- IBM 5410 Processing Unit
- IBM 5424 Multi-Function Card Unit (MFCU)
- IBM 5444 Disk Storage Drive
- IBM 1442 Model 6 Card Read Punch (with read column binary feature)

Section 2. Method of Operation

The 80-96 Conversion program first calls in a Program Protect transient routine to check for copyright violation.

Next the program halts displaying CU to allow the operator to set the Address/Data switch. The rightmost Address/Data switch is then checked to determine whether the destination cards are to be interpreted. If interpreting is not specified, the MFCU DTF is modified from punch/print to punch only.

The conversion language specifications are then read and checked for validity. These specifications are used to build the elements of the Conversion Table.

If the conversion language specifications contain no errors, the Conversion Table is built, and the source data is processed. A source card is read. The conversion for each element in the Conversion Table is then done. The Character Table is used by the A, C, and T conversion codes at this time to find the EBCDIC characters that match the punch pattern of the source card. When the end of the Conversion Table is reached, the output build area is moved to the output buffers and the destination card is punched.

Figure 7-1 shows the functional flow of the 80-96 Conversion program.

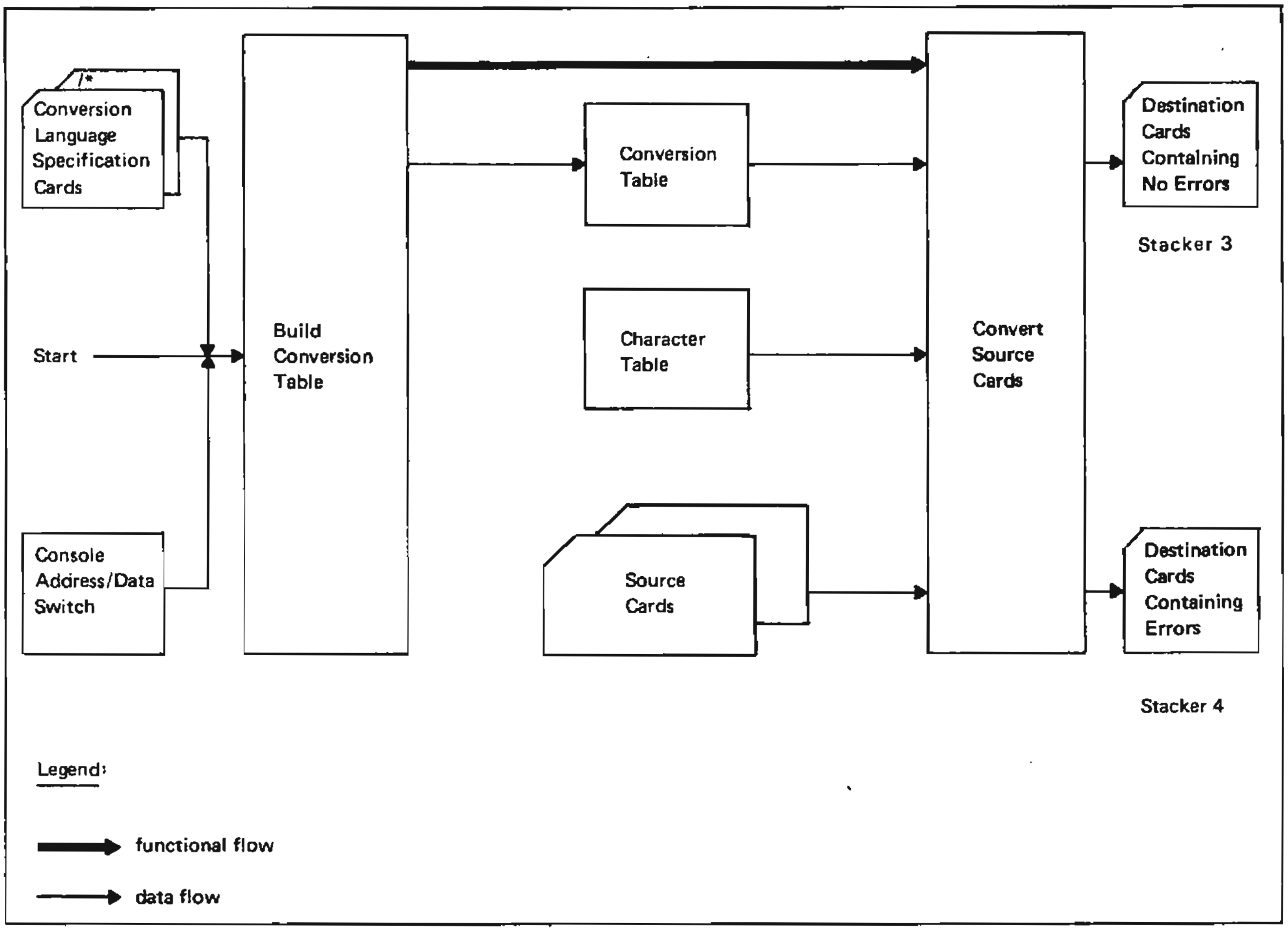


Figure 7-1. Functional Flow of Data and Control for 80-96 Conversion Program

Section 3. Program Organization

The two major functions of the 80-96 Conversion program are to:

- Build the Conversion Table from the conversion language specifications.

- Convert 80-column cards to 96-column cards.

This section discusses these functions in detail. A storage map of the program is shown in Figure 7-2. Chart JA is a flowchart of the program.

Program Boundary
Full Function IOS (MFCU and 1442)
MFCU Punch Buffer
MFCU Read Buffer/Build Area
Copyright
MFCU Print Buffer
1442 Read Buffer
1442 Read Column Binary
80-96 Conversion Program
Character Table
Conversion Table

Figure 7-2. Storage Map of the 80-96 Conversion Program

Building the Conversion Table

The first conversion language specification card is read. The first three entries (column 1-6) on the specification card are checked for validity. Valid entries for the source card (columns 1-4) are 01-80; valid entries for the destination card (columns 5-6) are 01-96. If an entry is invalid, a halt occurs. If an entry is valid, the entry is converted from decimal to a binary byte and stored in the first bytes of an element in the Conversion Table.

After the first three entries are processed, the conversion code is checked. If the code is invalid, a halt occurs. If the code is valid, it is added to the element in the Conversion Table.

Depending on the conversion code specified, information for the rest of the element in the table is then calculated and stored (see *Section 4, Data Area Formats* for the contents of the elements in this table for the various conversion codes).

When consecutive specifications have either all B conversion codes or all C conversion codes and the same entries in the source and destination columns, multiple substitutions are specified. Whenever a B or C conversion code is encountered, the source and destination card column entries are compared to the same entries in the previous element to find if entries in both are equal. If so, the number of substitutions in byte 5 of the Conversion Table element is increased by one and an additional entry is added to the end of the element.

One table element is built for each specification card read except for multiple substitutions. If the Conversion Table becomes too large for storage, the program halts. When end-of-file is reached on the specification cards, the address of the end of the Conversion Table is stored.

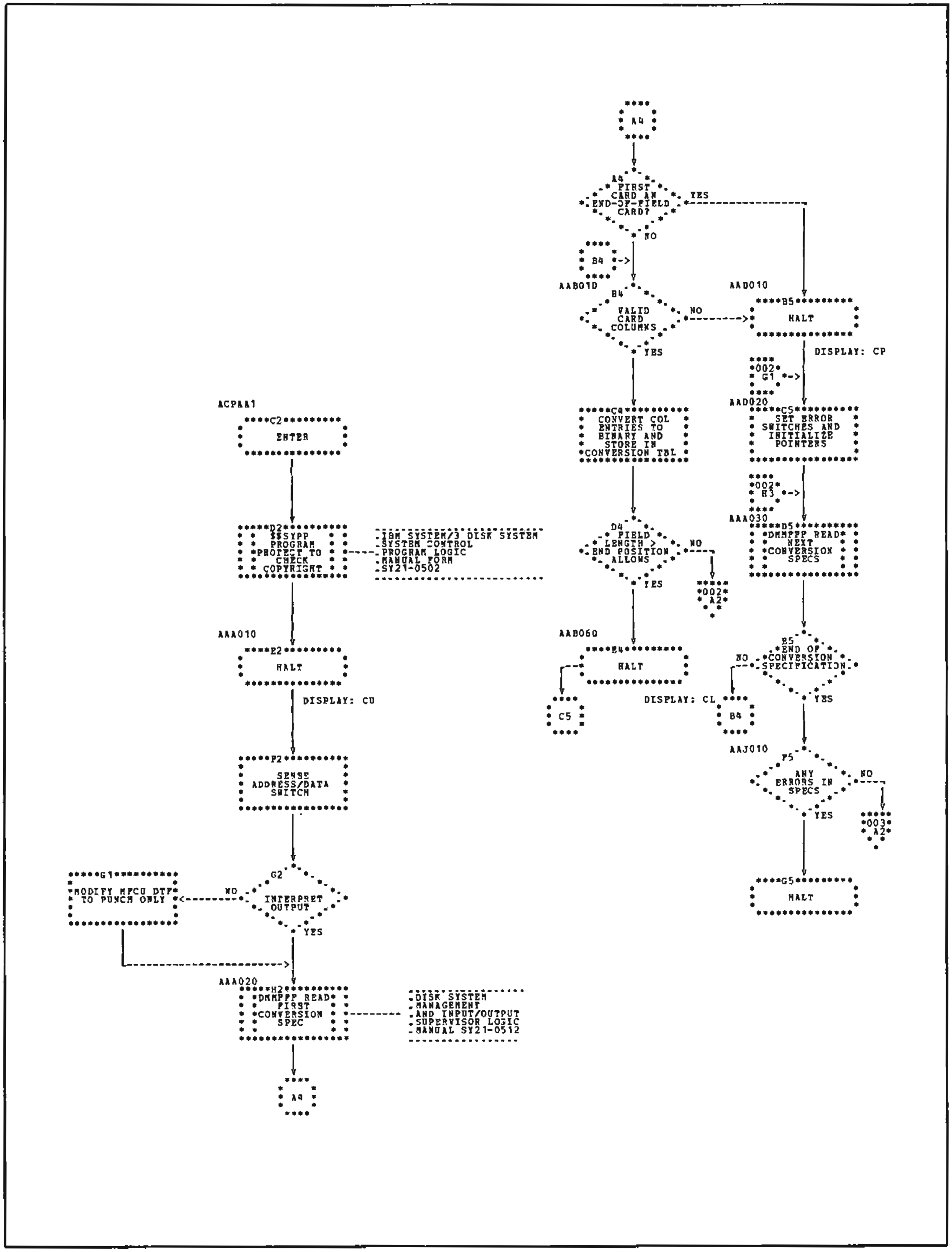

A4
A4
FIRST CARD AN END-OF-FIELD CARD?
YES
NO
B4
AAB010
B4
VALID CARD COLUMNS
NO
YES
AAD010
B5
HALT
DISPLAY: CP
002
G1
AAD020
C5
SET ERROR SWITCHES AND INITIALIZE POINTERS
002
H3
AAA030
D5
DMMPPP READ NEXT CONVERSION SPECS
C4
CONVERT COL ENTRIES TO BINARY AND STORE IN CONVERSION TBL
D4
FIELD LENGTH > END POSITION ALLOWS
NO
YES
002
A2
AAB060
E4
HALT
DISPLAY: CL
C5
B4
E5
END OF CONVERSION SPECIFICATION
NO
YES
AAJ010
F5
ANY ERRORS IN SPECS
NO
YES
003
A2
G5
HALT
ACPAA1
C2
ENTER
D2
$$SYPP PROGRAM PROTECT TO CHECK COPYRIGHT
IBM SYSTEM/3 DISK SYSTEM SYSTEM CONTROL PROGRAM LOGIC MANUAL FORM SY21-0502
AAA010
E2
HALT
DISPLAY: CU
F2
SENSE ADDRESS/DATA SWITCH
G1
MODIFY MFCU DTF TO PUNCH ONLY
G2
INTERPRET OUTPUT
NO
YES
AAA020
H2
DMMPPP READ FIRST CONVERSION SPEC
DISK SYSTEM MANAGEMENT AND INPUT/OUTPUT SUPERVISOR LOGIC MANUAL SY21-0512
A4

Chart JA. 80-96 Conversion Program (Part 1 of 8)

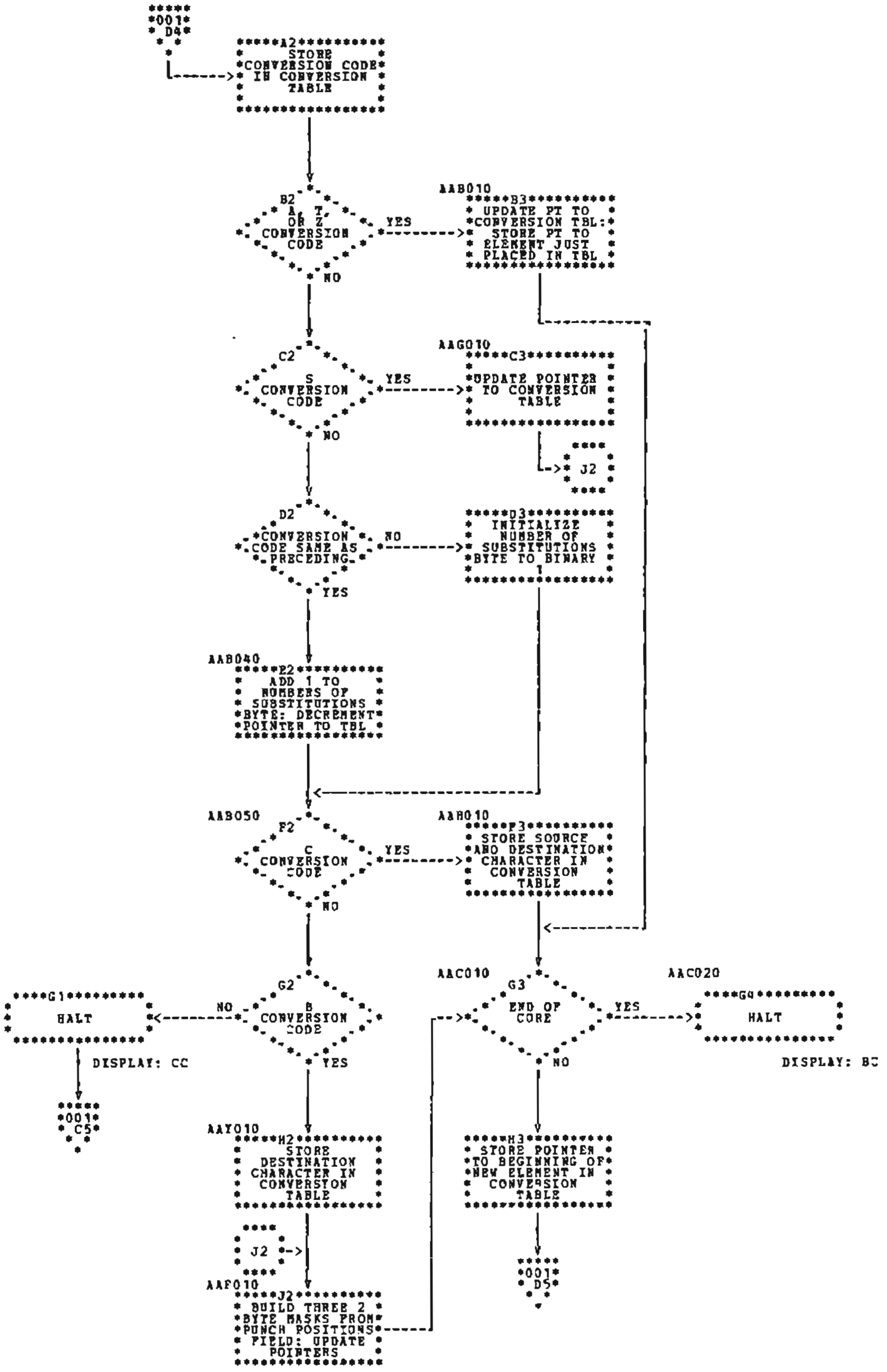

001
D4
A2
STORE CONVERSION CODE IN CONVERSION TABLE
B2
A, T, OR Z CONVERSION CODE
YES
NO
AAB010
B3
UPDATE PT TO CONVERSION TBL: STORE PT TO ELEMENT JUST PLACED IN TBL
C2
S CONVERSION CODE
YES
NO
AAG010
C3
UPDATE POINTER TO CONVERSION TABLE
J2
D2
CONVERSION CODE SAME AS PRECEDING
NO
YES
D3
INITIALIZE NUMBER OF SUBSTITUTIONS BYTE TO BINARY 1
AAB040
E2
ADD 1 TO NUMBERS OF SUBSTITUTIONS BYTE: DECREMENT POINTER TO TBL
AAB050
F2
C CONVERSION CODE
YES
NO
AAH010
F3
STORE SOURCE AND DESTINATION CHARACTER IN CONVERSION TABLE
G1
HALT
DISPLAY: CC
001
C5
G2
B CONVERSION CODE
NO
YES
AAC010
G3
END OF CORE
YES
NO
AAC020
G4
HALT
DISPLAY: BC
AAY010
H2
STORE DESTINATION CHARACTER IN CONVERSION TABLE
H3
STORE POINTER TO BEGINNING OF NEW ELEMENT IN CONVERSION TABLE
J2
001
D5
AAF010
J2
BUILD THREE 2 BYTE MASKS FROM PUNCH POSITIONS FIELD: UPDATE POINTERS

Chart JA. 80-96 Conversion Program (Part 2 of 8)

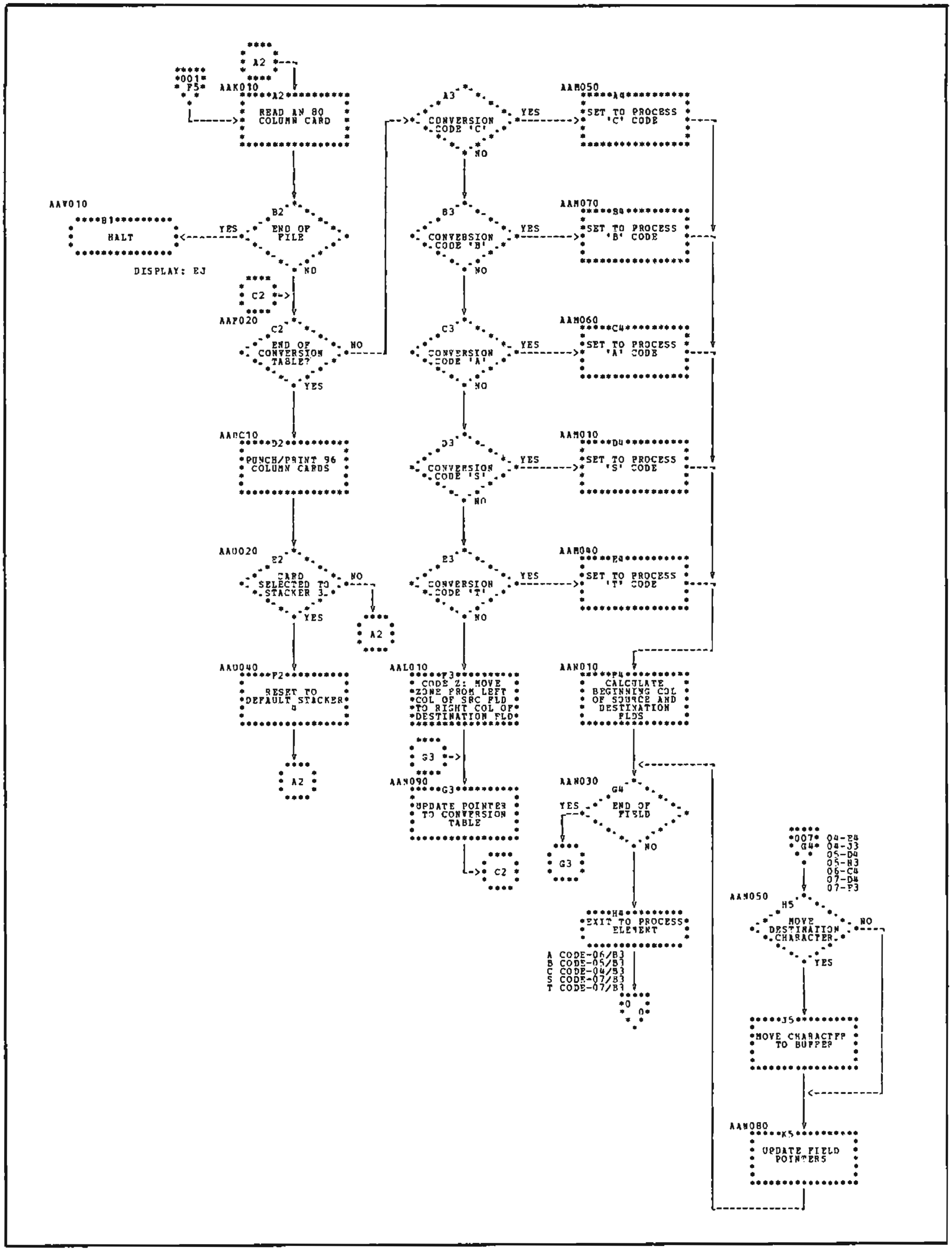


Chart JA. 80-96 Conversion Program (Part 3 of 8)

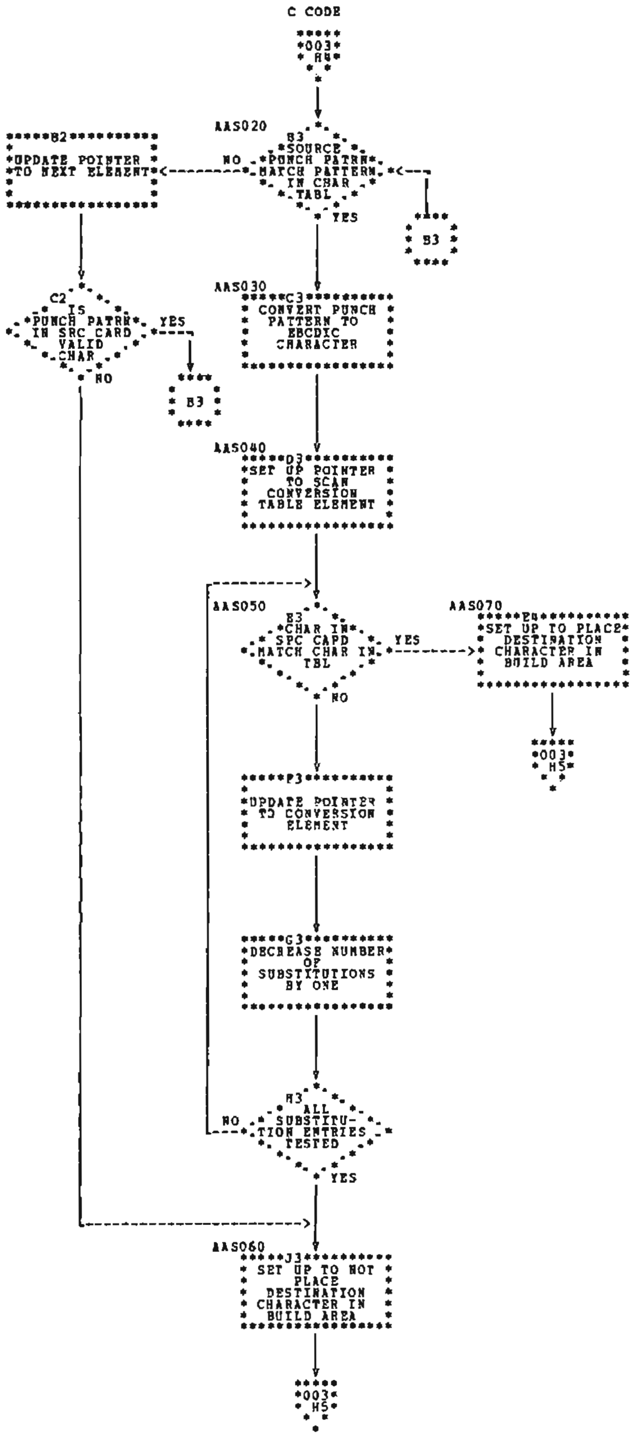

C CODE
003 H4
AAS020
B3
SOURCE PUNCH PATRN MATCH PATTERN IN CHAR TABL
NO
YES
B3
B2
UPDATE POINTER TO NEXT ELEMENT
C2
IS PUNCH PATRN IN SRC CARD VALID CHAR
YES
NO
B3
AAS030
C3
CONVERT PUNCH PATTERN TO EBCDIC CHARACTER
AAS040
D3
SET UP POINTER TO SCAN CONVERSION TABLE ELEMENT
AAS050
E3
CHAR IN SPC CAPD MATCH CHAR IN TBL
YES
NO
AAS070
E4
SET UP TO PLACE DESTINATION CHARACTER IN BUILD AREA
003 H5
F3
UPDATE POINTER TO CONVERSION ELEMENT
G3
DECREASE NUMBER OF SUBSTITUTIONS BY ONE
H3
ALL SUBSTITU-TION ENTRIES TESTED
NO
YES
AAS060
J3
SET UP TO NOT PLACE DESTINATION CHARACTER IN BUILD AREA
003 H5

Chart JA. 80-96 Conversion Program (Part 4 of 8)

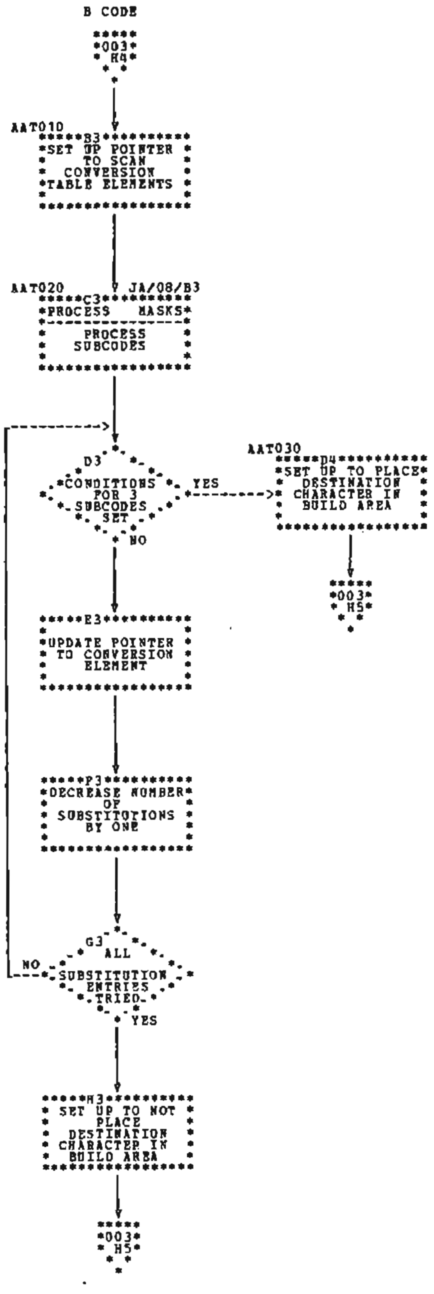

B CODE
003
H4
AAT010
B3
SET UP POINTER TO SCAN CONVERSION TABLE ELEMENTS
AAT020
JA/08/B3
C3
PROCESS MASKS
PROCESS SUBCODES
D3
CONDITIONS FOR 3 SUBCODES SET
YES
NO
AAT030
D4
SET UP TO PLACE DESTINATION CHARACTER IN BUILD AREA
003
H5
E3
UPDATE POINTER TO CONVERSION ELEMENT
F3
DECREASE NUMBER OF SUBSTITUTIONS BY ONE
G3
ALL SUBSTITUTION ENTRIES TRIED
NO
YES
H3
SET UP TO NOT PLACE DESTINATION CHARACTER IN BUILD AREA
003
H5

Chart JA. 80-96 Conversion Program (Part 5 of 8)

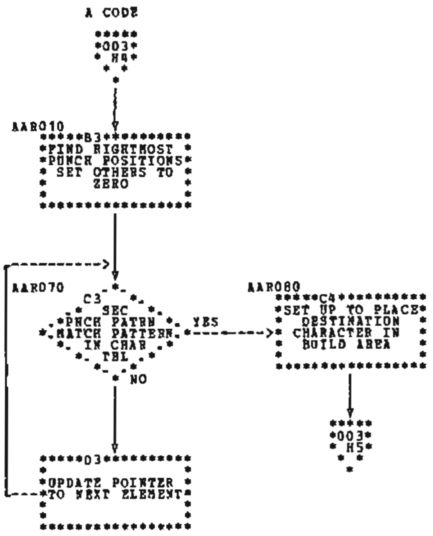

A CODE
003
H4
AAR010
B3
FIND RIGHTMOST PUNCH POSITIONS SET OTHERS TO ZERO
AAR070
C3
SEC PNCH PATRN MATCH PATTERN IN CHAR TBL
YES
NO
AAR080
C4
SET UP TO PLACE DESTINATION CHARACTER IN BUILD AREA
003
H5
D3
UPDATE POINTER TO NEXT ELEMENT

Chart JA. 80-96 Conversion Program (Part 6 of 8)

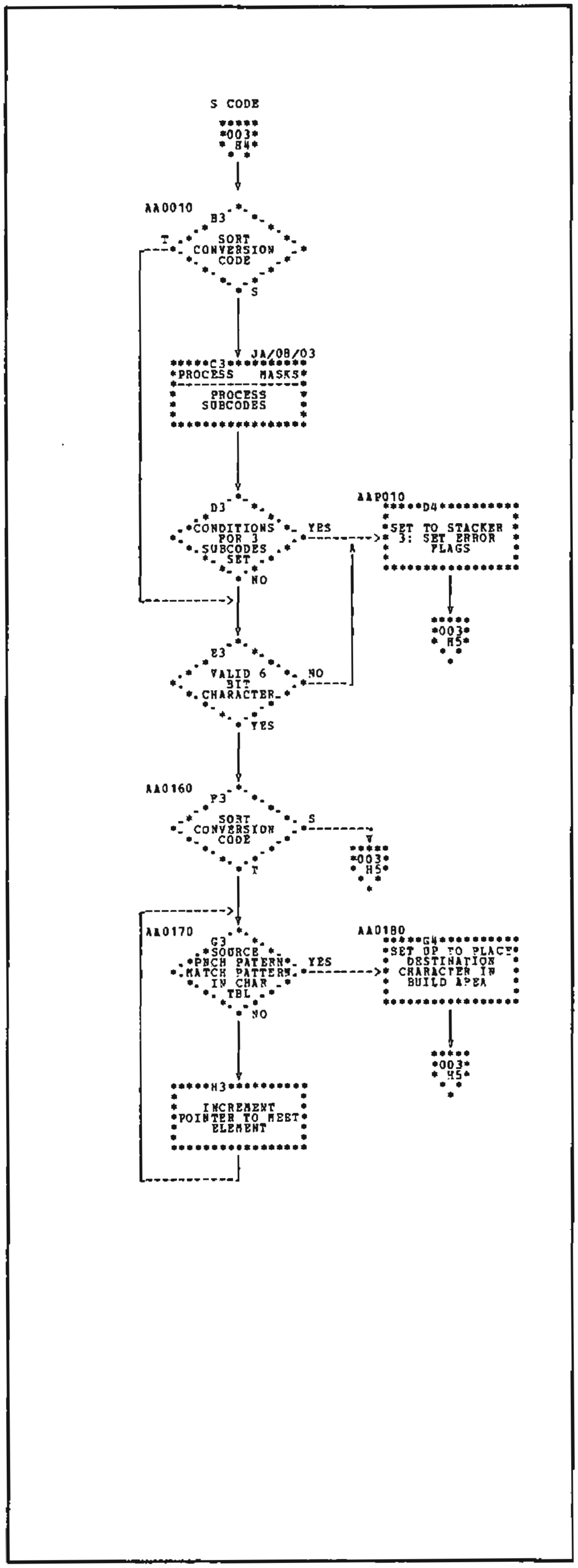


Chart JA. 80-96 Conversion Program (Part 7 of 8)

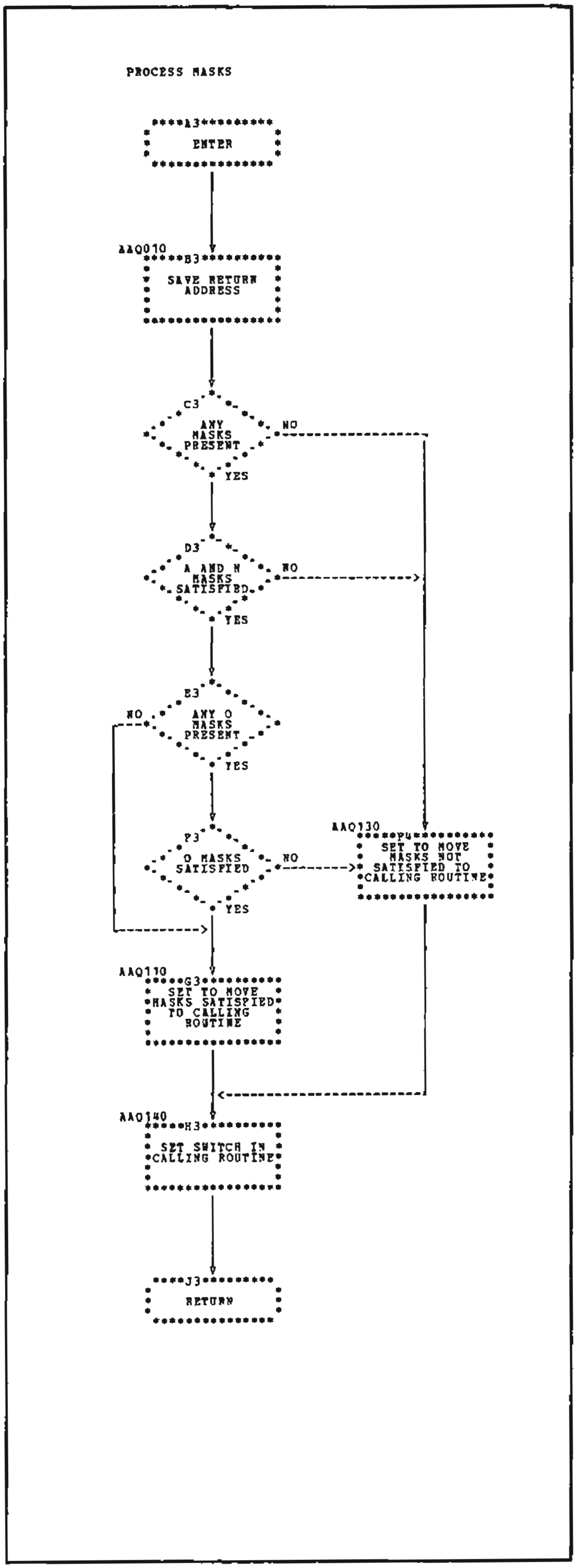


Chart JA. 80-96 Conversion Program (Part 8 of 8)

Conversion Table

The Conversion Table is built from the conversion language specifications at execution time. The number of elements in the table is dependent on the number of conversion language specifications. The number of bytes in each element is dependent on the particular conversion code of the specification. Figure 7-3 shows the contents of an element for each conversion code.

Character Table

The Character Table contains 64 elements, one element for each of the characters valid for a 96-column card.

Each element is described in four bytes in the following format:

Byte	*Bit*	*Contents*
Source card image punches:		
1	0-1	Not used
	2	Twelve punch position
	3	Eleven punch position
	4	Zero punch position
	5	One punch position
	6	Two punch position
	7	Three punch position
2	0-1	Not used
	2	Four punch position
	3	Five punch position
	4	Six punch position
	5	Seven punch position
	6	Eight punch position
	7	Nine punch position
Destination card image punches:		
3	0-1	Not used
	2	B punch position
	3	A punch position
	4	8 punch position
	5	4 punch position
	6	2 punch position
	7	1 punch position
	0-7	Hexadecimal representation of previous punches

Copyright

This 46-byte area contains the program number for this program and copyright information as follows: 5702-UT1ƀCOPYRIGHTƀIBMƀCORPƀ1970. (The ƀ represents a blank.) The remainder of the area is filled with blanks.

Bytes	B Code	C Code	S Code	A Code	T Code	Z Code
1	Beginning source card column (in binary)					
2	Ending source card column (in binary)					
3	Ending destination card column (in binary)					
4	Conversion code character B	C	S	A	T	Z
5	Number of substitutions for source card		Mask for subcode A			
6	Mask for subcode A	Character in the source card column				
7		Character to be punched into destination card column	Mask for subcode N			
8	Mask for subcode N	Character in the source card column				
9		Character to be punched into destination card column	Mask for subcode O			
10	Mask for subcode O					
11						
12	Character to be punched into destination card column					
13	Mask for subcode A					
14						
15	Mask for subcode N					
16						
17	Mask for subcode O					
18						
19	Character to be punched into destination card column					

Note 1: The length of the element for conversion codes B and C depends on the number of substitutions specified (byte 5). The entries below the heavy black lines on the chart indicate the information that will be repeated for each substitution.

Note 2: The 2-byte masks for the subcodes have this format:

Byte	Bit	Contents
First	0-1	Not used
	2	Twelve punch position
	3	Eleven punch position
	4	Zero punch position
	5	One punch position
	6	Two punch position
	7	Three punch position
Second	0-1	Not used
	2	Four punch position
	3	Five punch position
	4	Six punch position
	5	Seven punch position
	6	Eight punch position
	7	Nine punch position

Figure 7-3. Conversion Table Elements

Appendix: Directory

The directory lists each of the utility programs for reference to the program listings on microfiche.

Descriptive Name	*Entry Point*	*Synopsis*
96-List	LSLIST	Reads, counts, and prints a listing of 96-column cards.
96-96 Reproduce and Interpret	REPRO	Reproduces, interprets, and reformats 96-column cards.
Sort/Collate – Generation	ASMAA1	Reads and diagnoses specification cards, prints a source listing, and generates object code.
Sort/Collate -- Diagnostics Print	ASMAB1	Diagnoses header card, prints error messages, and sets switches for selection of job module.
Sort/Collate -- Execution	ASMAC1 ASMAD1 ASMAE1 ASMAF1	Selects the specified job module: Sort -- ASMAC1, Merge -- ASMAD1, Match -- ASMAE1, Select – ASMAF1; processes data according to the generated code.
Gangpunch Diagnostic Phase	GANGP	Reads and diagnoses the header record and the field definition record(s); builds the FDP table and stores information in the common area that will be used by the execution phase prints all error messages; cancels the job if terminal errors are diagnosed; gives control to the execution phase.
Gangpunch Execution Phase	GPEXC	Gangpunches detail records according to the header and field definition records processed in the diagnostic phase.
Data Recording	ADRAA1	Simulates the IBM 5496 Data Recorder; accepts control cards which specify the format of the cards to be punched.
Data Verifying	ADVAA1	Simulates the IBM 5496 Data Recorder; operating in the verify mode; accepts control cards which specify the format of the card image during verification.
80-96 Conversion	ACPAA1	Converts the contents of an 80-column card to 96-column card.

96-List

address/data switch settings 1-2

copyright 1-4

data/address switch settings 1-2
data areas 1-4

end of file 1-1

flowchart 1-3
functions 1-1

Halt/Syslog message table 1-4

LSBF1 1-4
LSBF2 1-4

read area/work buffer (LSBF1, LSBF2) 1-4

storage map 1-2
switch settings, address/data 1-2
system requirements 1-1

work buffer/read area (LSBF1, LSBF2) 1-4

96-96 Reproduce and Interpret

address/data switch settings 2-3

basic reproducing functions 2-2
buffer associated IOBs (RDIB11, RPCIB1) 2-6

copyright 2-6

data/address switch settings 2-3
data areas 2-6
define the file (DTF) 2-6
DTF (define the file) 2-6

flowchart 2-4
functions 2-1

Halt/Syslog message table (RHLTB) 2-6

interpret only 2-2
interpreting with reproducing function 2-2
IOBs, buffer associated (RDIB1, RPCIB1) 2-6

logical record area (RBFOUT) 2-6

print I/O area (RPRTIO) 2-6
punch I/O area (RPCIO1) 2-6

RBFOUT (logical record area) 2-6
RDIB1 (buffer associated IOB) 2-6
RDIO1 (read I/O area) 2-6
read I/O area (RDIO1) 2-6
reformat table (RTABL) 2-6
reformatting functions 2-2
reproduction functions
  basic reproducing 2-2
  interpreting with reproducing 2-2
  reformatting 2-2
RHLTB (Halt/Syslog message table) 2-6
RPCIB1 (buffer associated IOB) 2-6
RPCIO1 (punch I/O area) 2-6
RPRTIO (print I/O area) 2-6
RTABL (reformat table) 2-6
RWORK (save area) 2-6

save area (RWORK) 2-6
storage map 2-3
switch settings, address/data 2-3
system requirements 2-1

Sort/Collate

abbreviations, code segment instructions 3-13
address table, control word 3-20
alternate collating sequence table 3-4

beginning of force lines code segment 3-19
beginning of set code segment 3-13
beginning of subset code segment 3-13
branch instruction code segment 3-17
branch on condition instruction code segment 3-16
branch to include/omit code segment 3-20
branch to job module code segment 3-13
build control word, description 3-12

character-field to constant code segment 3-14
character-field to field code segment 3-14
character code segment (part 1), forced field 3-19
character code segment (part 2), forced field 3-19
character code segment, normal field 3-17

code, generated 3-12
code segment instruction abbreviations 3-13
code segments
  branch to job module 3-13
  control word 3-17
  description 3-12
  flowchart 3-39
  record identification 3-13
conditional branch instruction code segment 3-16
control field card code segments (see control word code segments)
control word address table 3-20
control word build, description 3-12
control word code segments 3-17
constant, jump over
  control word 3-18
  record identification 3-14
copyright 3-7

data areas
  control word address table 3-20
  copyright 3-7
  sort/collate interphase area 3-7
data flow
  diagnostic print phase 3-3
  execution phase 3-10
  generation phase 3-2
diagnostics error message print phase (see diagnostics print phase)
diagnostics print phase
  detailed description 3-6
  flow of logic 3-3
  functions 3-1
digit – field to constant code segment 3-15
digit – field to field code segment 3-15
digit code segment (part 1), forced field 3-19
digit code segment (part 2), forced field 3-19
digit code segment, normal field 3-18
digit code segment, opposite field 3-18
dump analysis, sample 3-21

error table 3-5
execution phase
  functions 3-1
  object program 3-10

field to constant code segment, character 3-14
field to constant code segment, digit 3-15
field to constant code segment, unpacked 3-15
field to field code segment, character 3-14
field to field code segment, digit 3-15
field to field code segment, unpacked 3-15
force-all code segment 3-19
force lines code segment, beginning of 3-19
force sequence – leading instruction code segment 3-18
forced field – character code segment (part 1) 3-19
forced field – character, zone, digit code segment (part 2) 3-19
forced field – digit code segment (part 1) 3-19
forced field – zone code segment (part 1) 3-19
functions 3-1

generated code 3-12
generation phase
  detailed description 3-4
  flow of logic 3-2
  functions 3-1

include/omit code segment, branch to 3-20
INO branch
  beginning of a subset code segment 3-13
  branch on condition instruction code segment 3-15
  set/subset description 3-12
input/output flow
  diagnostic print phase 3-3
  execution phase 3-10
  generation phase 3-2
interphase area, sort/collate 3-7
IYES branch
  beginning of a set code segment 3-13
  branch on condition instruction code segment 3-15
  set/subset description 3-12

job module, branch to 3-13
job modules
  match job module 3-11
  merge job module 3-11
  select job module 3-11
  sort job module 3-10
jump over constant code segment
  control word 3-18
  record identification 3-14

leading instruction code segment, force sequence 3-18

match job module
  description 3-11
  flowchart 3-33
merge job module
  description 3-11
  flowchart 3-32

normal field – character code segment 3-17
normal field – digit code segment 3-18
normal field – unpacked code segment 3-18
normal field – zone code segment 3-17

object program
  code segments 3-12
  job modules 3-10
omit/include code segment, branch to 3-20
opposite field – digit code segment 3-18
opposite field – unpacked code segment 3-18

record identification code segments 3-13
record identification logic, description 3-12

sample dump analysis 3-21
segments, code (see code segments)
select job module
  description 3-11
  flowchart 3-34
set, record identfication logic 3-12
set code segment, beginning of 3-13
set stacker for stacker select code segment 3-14
sort/collate generation and diagnostic phase (see generation phase)
sort/collate interphase area 3-7
sort job module
  description 3-10
  flowchart 3-31
stacker select code segment, set stacker for 3-14
storage map
  execution phase 3-10
  generation and diagnostics print phase 3-4
subset, record identification logic 3-12
subset code segment, beginning of 3-13
system requirements 3-1

unpacked – field to constant 3-15
unpacked – field to field 3-15
unpacked field – normal or opposite 3-18

zone code segment 3-14
zone code segment (part 1), forced field 3-19
zone code segment (part 2), forced field 3-19
zone code segment, normal field 3-18

Gangpunch

buffers
  physical print (PPNTBF) 4-33
  physical punch (PPCHBF) 4-33
  physical read (PREDBF) 4-33
  read (GPRD1) 4-33

common area 4-22
count controlled gangpunch
  description 4-1
  diagram 4-8
  function 4-2

data areas 4-22
diagnostic phase
  description 4-1, 4-9
  diagrams 4-3
    process header record 4-4
    process field definition record 4-5
  flowchart 4-11
  routines used 4-13, 4-14
  storage map 4-9
DTF (PRIDTF and SECDTF) 4-29

error table 1 (ERTAB1) 4-30
ERTAB1 (error table 1) 4-30
execution phase
  description 4-1, 4-9
  flowchart 4-16
  routines used 4-19 – 4-21
  storage map 4-9

FDP table (FDPADS) 4-31
FDPADS (FDP table) 4-31
field definition record processing
  diagram 4-5

GRPD1 (read buffer/work area) 4-33

header record processing diagram 4-4

interspersed gangpunch
  description 4-1
  diagram 4-6
  function 4-2

logical record buffer (LRBPRT) 4-33
LRBPRT (logical record buffer) 4-33

match field gangpunch
  description 4-1
  diagram 4-7
  function 4-2

overview diagram 4-3

PPCHBF (physical punch buffer) 4-33
PPNTBF (physical print buffer) 4-33
PRIDTF (define the file) 4-29
PREDBF (physical read buffer) 4-33
program structure 4-1

routines, diagnostic phase
  decimal to binary conversion 4-13
  master or detail record selector information check 4-13
  numeric field test 4-13
  print a record 4-14
  print error messages 4-14
routines, execution phase
  build the output record 4-20
  determine the record type 4-20
  I/O interface 4-21
  reset the counter 4-19
  sequence check the match field 4-19
  update the counter 4-19

SECDTF (define the file) 4-29
storage map 4-9
system requirements 4-1

table of contents, visual 4-2

work area (GPRD1) 4-33

Data Recording

adjust routine
  description 5-9
  flowchart 5-20
assembly area (ASSM) 5-22
ASSM (assembly area) 5-22

BCNT (binary column indicator) 5-22
beginning of end of program defined field routine description 5-9
binary column indicator (BCNT) 5-22
buffer associated IOB (PUIOB, RDIOB) 5-26

CCODE (control code) 5-24
column indicator (LITE) 5-23
control code (CCODE) 5-24
copyright 5-26
current control area (PROG) 5-24

data area activity 5-28
data areas 5-22
data flow 5-2
data key/self-check routine
  description 5-6
  flowchart 5-10
DATA/STAT (sense table) 5-24
DCNT (decimal column indicator) 5-22
decimal column indicator (DCNT) 5-22
define the file (DTF) 5-26
display routine, description 5-4
DTF (define the file) 5-26
dup key routine
  description 5-7
  flowchart 5-13

end or beginning of program defined field routine, description 5-9
error reset key routine, description 5-8

field erase key routine
  description 5-7
  flowchart 5-16
functions 5-1

HOLD (hold area) 5-22
hold area (HOLD) 5-22

initializing routine, description 5-4
input/output flow 5-2
interrupt handler routine, description 5-4
interrupt service routine, description 5-6
interrupts 5-2
invalid character routine, description 5-7
IOB, buffer associated (PUIOB, RDIOB) 5-26
IOCS parameter list, MFCU 5-26
IOS interface routine
  description 5-9
  flowchart 5-21

LITE (column indicator) 5-23

MFCU IOCS parameter list 5-26

parameter list, MFCU IOCS 5-26
PGM1 (program 1) 5-24
PGM2 (program 2) 5-24
print area
  PRINT0 5-22
  PRINT1 5-22
PRINT0 (print area) 5-22
PRINT1 (print area) 5-22
PROG (current control area) 5-24
program load switch routine
  description 5-9
  flowchart 5-19
program 1 (PGM1) 5-24
program 1 key routine
  description 5-8
  flowchart 5-17
program 2 (PGM2) 5-24
program 2 key routine
  description 5-18
  flowchart 5-18
PUIOB (buffer associated IOB) 5-26

RDBUF (read buffer) 5-26
RDIOB (buffer associated IOB) 5-26
read buffer (RDBUF) 5-26
read key routine, description 5-8
record erase switch routine, description 5-8
record release switch routine, description 5-8
release key routine
  description 5-7
  flowchart 5-15
return routine, description 5-4
right adjust key routine
  description 5-7
  flowchart 5-14

self check/data key routine
  description 5-6
  flowchart 5-10
sense table (STAT/DATA) 5-24
skip key routine
  description 5-7
  flowchart 5-12
STAT/DATA (sense table) 5-24
stick-light table (TAB) 5-23
storage map 5-6
system requirements 5-1

TAB (stick-light table) 5-23
test for auto skip field routine, description 5-9

Data Verifying

adjust routine
  description 6-10
  flowchart 6-23
assembly area (ASSM) 6-26
ASSM (assembly area) 6-26

BCNT (binary column indicator) 6-26
beginning or end of program defined field routine description 6-10
binary column indicator (BCNT) 6-26
buffer associated IOB (PUIOB, RDIOB) 6-30

CCODE (control code) 6-29
column indicator (LITE) 6-26
column indicator display routine, description 6-5
control coe (CCODE) 6-29
copyright 6-30
correct area (CORRT) 6-26
CORRT (correct area) 6-26
current control area (PROG) 6-28

data area activity 6-30
data areas 6-26
data flow 6-2
data key routine
  description 6-6
  flowchart 6-11
DATA/STAT (sense table) 6-28
DCNT (decimal column indicator) 6-26
decimal column indicator (DCNT) 6-26
define the file (DTF) 6-30
DTF (define the file) 6-30
DUP key routine
  description 6-7
  flowchart 6-13

end or beginning of program defined field routine, description 6-10
error code routine, description 6-7
error reset key routine, description 6-8

field erase key routine
  description 6-8
  flowchart 6-19
flow of data 6-2
functions 6-1

HOLD (hold area) 6-26
hold area (HOLD) 6-26

initializing routine, description 6-5
input/output flow 6-2
interrupt handler routine, description 6-6
interrupt service routine, description 6-6
interrupts 6-2
invalid character routine, description 6-7
IOB (MFCU IOCS parameters) 6-30
IOB, buffer associated (PUIOB, RDIOB) 6-30
IOCS parameters, IOB 6-30
IOS interface routine
  description 6-10
  flowchart 6-25

LITE (column indicator) 6-26

MFCU IOCS parameters (IOB) 6-30

parameters, MFCU IOCS 6-30
PGM1 (program 1) 6-28
PGM2 (program 2) 6-28
print area (PRINT1) 6-30
print 2 area (PRT2) 6-30
PRINT1 (print area) 6-30
PROG (current control area) 6-28
program load switch routine
  description 6-9
  flowchart 6-22
program 1 (PGM1) 6-28
program 1 key routine
  description 6-9
  flowchart 6-20
program 2 (PGM2) 6-28
program 2 key routine
  description 6-9
  flowchart 6-21
PRT2 (print 2 area) 6-30
PUIOB (buffer associated IOB) 6-30

RDIOB (buffer associated IOB) 6-30
read key routine, description 6-9
record erase switch routine, description 6-9
record release switch routine, description 6-9
release key routine
  description 6-8
  flowchart 6-15
REM (remember switch) 6-30
remember switch (REM) 6-30
return routine, description 6-6
right adjust key routine
  description 6-8
  flowchart 6-14

sense table (STAT/DATA) 6-28
skip key routine
  description 6-7
  flowchart 6-12
STAT/DATA (sense table) 6-28
stick-light table (TAB) 6-26
storage map 6-5
system requirements 6-1

TAB (stick-light table) 6-26
test for auto skip field routine, description 6-10

80-96 Conversion

character table 7-10
conversion process 7-3
conversion table
  building 7-3
  format 7-10
copyright 7-10

data areas
  character table 7-10
  conversion table 7-10
  copyright 7-10
data flow 7-2

flowchart 7-4
functions 7-1

input/output flow 7-2

storage map 7-3
system requirements 7-1

LY21-0523-1

IBM System/3 Printed in U.S.A. LY21-0523-1

IBM

International Business Machines Corporation
Data Processing Division
1133 Westchester Avenue, White Plains, New York 10604
(U.S.A. only)

IBM World Trade Corporation
821 United Nations Plaza, New York, New York 10017
(International)