IBM SYSTEM/3

IBM System/3 Card System System Control Programs Logic Manual

Program Number 5701-SC1

Preface

This Program Logic Manual is designed to satisfy the documentation requirements of support personnel responsible for maintenance of these IBM System/3 programs:

- System Initialization.
- Program Maintenance.
- Device Counter Logout.

Information in this manual includes both general and detailed descriptions of each of the three programs.

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,* GA21-9103.
- *IBM System/3 Card System: Operator's Guide,* GC21-7513.
- *IBM System/3 Card System: Absolute Card Loader and Input/Output Control System Logic Manual,* SY21-0521.

Third Edition (January 1971)

This is a major revision of, and obsoletes, SY21-0522-1 and Technical Newsletters SN21-7517 and SN21-7530. Changes have been made to the System Initialization program and a new program, the Device Counter Logout program, has been added. Changes to the text and small changes to illustrations are indicated by a vertical line at the left of the change; changed or added illustrations are denoted by the symbol ● to the left of the caption.

This edition applies to version 1, modification 4, of Program Product 5701-SC1 and to all subsequent versions and modifications unitl otherwise indicated in new editions or Technical Newsletters. Changes are continually made to the specifications herein; before using this publication in connection with the operation of IBM Systems, consult the latest IBM System/3 Newsletter, Order Number GN20-2228, for the editions that are applicable and current.

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.

© Copyright International Business Machines Corporation 1970, 1971

Contents

HOW THIS PUBLICATION IS ORGANIZED

This publication is divided into three parts—one part for each program. The following sections are included, as required, in each part depending on the size and complexity of the program being described:

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

- *Method of Operation* describes the data flow and functional flow of the program in general terms.

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

- *Data Area Formats* describes significant data areas used by each program.

Two appendixes in the back of this publication contain information applicable to all programs:

- *Appendix A* is a directory containing information needed for quick reference to the program listings.

- *Appendix B* discusses the flowcharting techniques used in this publication.

Section 1. Introduction

The IBM System/3 System Initialization program initializes storage for the system user. The program performs these functions:

- Punches a card containing the date and information recorded in the Error Recording area. The program inserts this card before the end card of the first module.
- Initializes the Error Recording area to binary zeros.
- Initializes the copyright area to blanks.
- Initializes storage to blanks from the first byte following the System Initialization program to the highest storage position (according to the system storage size).
- Reads, identifies, and interprets control cards.
    1. Builds a chain image in storage using the chain image control cards.
    2. Loads the system date specified on the date control cards.
    3. Loads the system storage size specified on the storage size control card.
    4. Sets external indicators as specified on the external indicator control card.

Control Cards

The System Initialization program accepts four control cards.

- Chain image control card.
- Date control card.
- External indicator control card.
- Storage size control card.

Punched Card

A card is punched by the System Initialization program and inserted before the end card of the first module. The contents of this card are:

Column	*Contents*
1	W
2-65	Q, R, and sense bytes from last eight I/O errors (location X'180'-X'19F')
66-77	Last six hammer echo checks (location X'1A0'-X'1A5')
78-93	Blanks
94	Month 1-9 = January to September 0 = October – (minus) = November } = December
95	Day A-I = 01-09 0̄-R = 10-19 8-9 = 20-21 S-Z = 22-29 1-2 = 30-31
96	Year (last digit of year)
123-128	Printed date from System Communication area (mmddyy)

Columns 94-96 will be blank if the information in storage is not a positive 6-digit number. Column 95 will be blank if the day is not 01-31.

See Section 3, *Data Area Formats*, for the information in the Error Recording area. For these areas, one byte in storage requires two columns on the punched card.

System Requirements

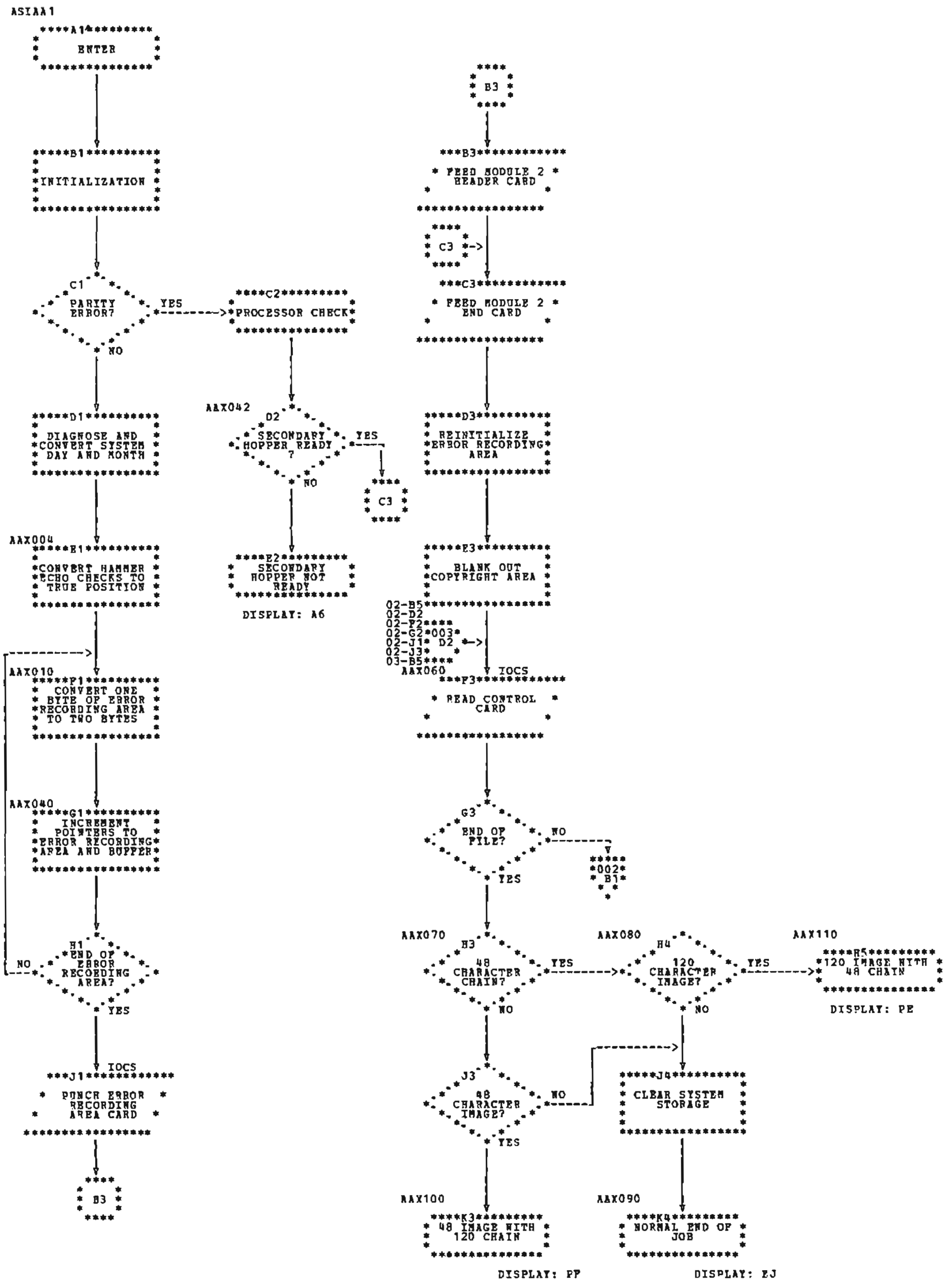
The System Initialization program will function with any configuration of the IBM System/3 Card System.

Section 2. Program Organization

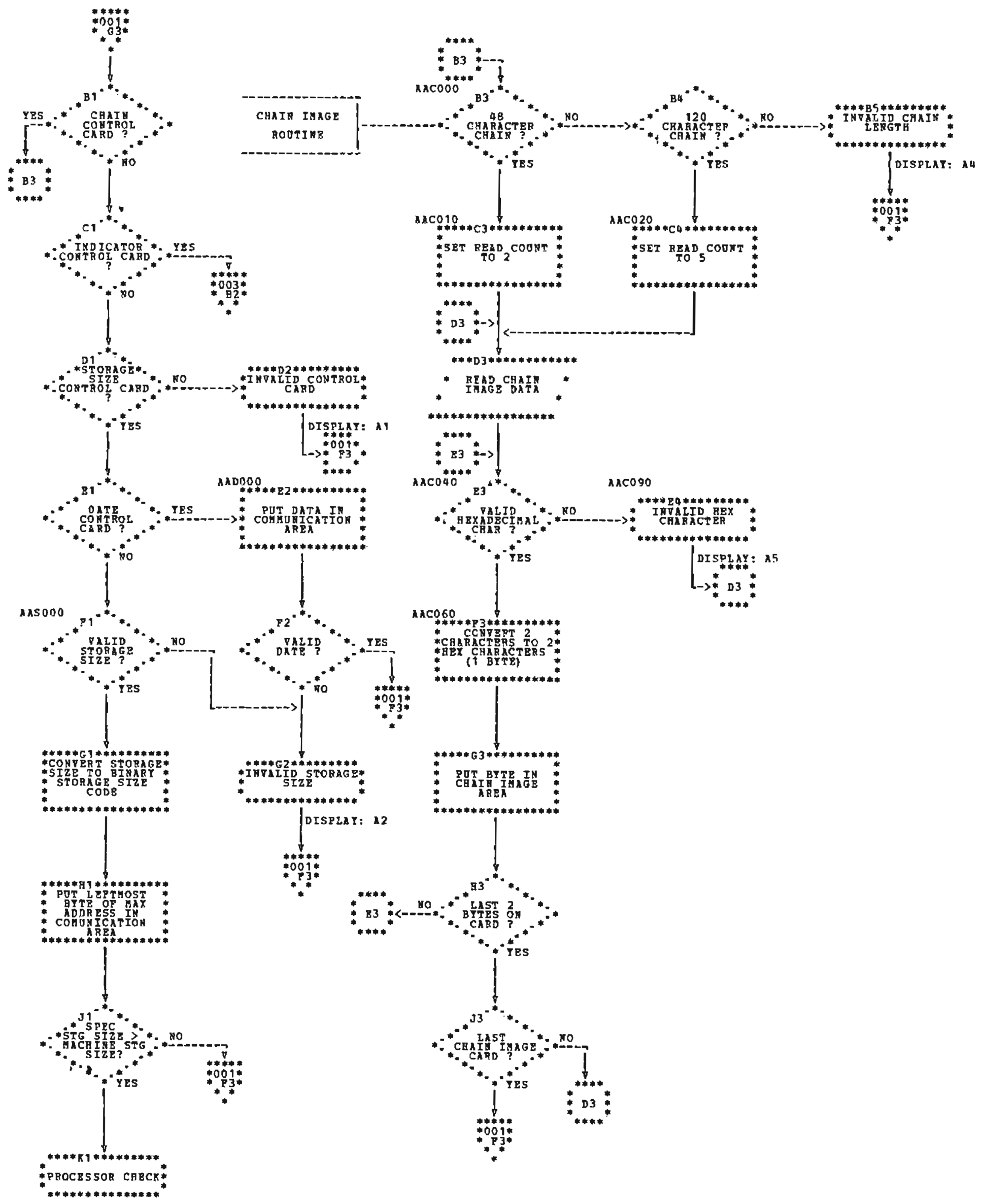
The System Initialization program is made up of two modules. The first module initializes the System Communication area. If a parity error processor check occurs while the first module is running, the user presses the Program Load key which loads the header card of the second module.

The second module consists of a header card and a dummy end card. The header card causes a branch back to the first module where the Error Recording area is re-initialized but not punched. This allows the System Initialization program to continue. If no processor check occurs, the second module is fed by the first module.

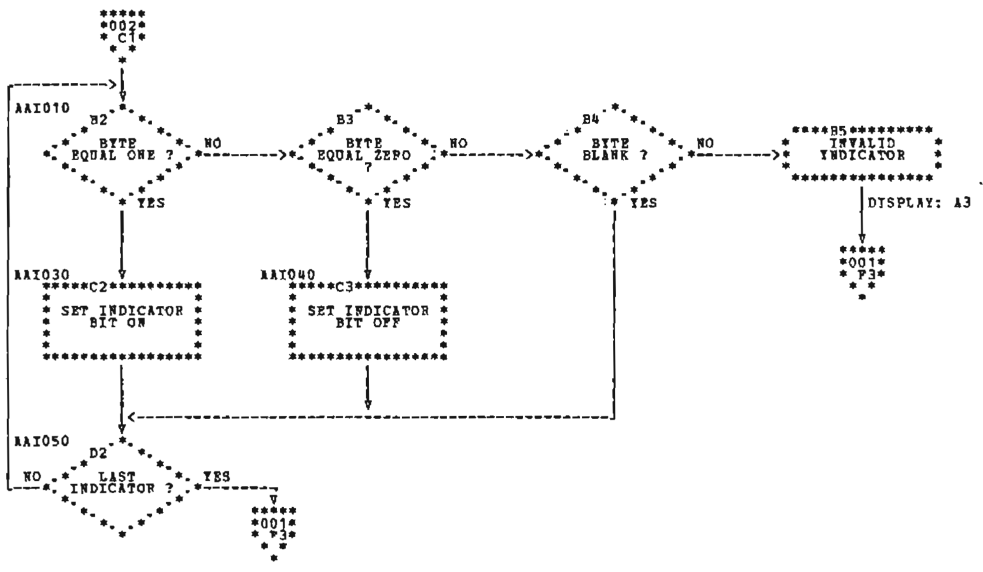
The end card is not used by the second module and is always fed by the first module. The flowchart for the System Initialization program is found on Chart BA (parts 1, 2, and 3).


ASIAA1
A1 ENTER
B1 INITIALIZATION
C1 PARITY ERROR?
YES
NO
C2 PROCESSOR CHECK
AAX042
D2 SECONDARY HOPPER READY ?
YES
C3
NO
E2 SECONDARY HOPPER NOT READY
DISPLAY: A6
D1 DIAGNOSE AND CONVERT SYSTEM DAY AND MONTH
AAX004
E1 CONVERT HAMMER ECHO CHECKS TO TRUE POSITION
AAX010
F1 CONVERT ONE BYTE OF ERROR RECORDING AREA TO TWO BYTES
AAX040
G1 INCREMENT POINTERS TO ERROR RECORDING AREA AND BUFFER
H1 END OF ERROR RECORDING AREA?
NO
YES
IOCS
J1 PUNCH ERROR RECORDING AREA CARD
B3
B3
B3 FEED MODULE 2 HEADER CARD
C3
C3 FEED MODULE 2 END CARD
D3 REINITIALIZE ERROR RECORDING AREA
E3 BLANK OUT COPYRIGHT AREA
02-B5
02-D2
02-F2
02-G2
02-J1
02-J3
03-B5
003
D2
AAX060
IOCS
F3 READ CONTROL CARD
G3 END OF FILE?
NO
002 B1
YES
AAX070
H3 48 CHARACTER CHAIN?
YES
AAX080
H4 120 CHARACTER IMAGE?
YES
AAX110
H5 120 IMAGE WITH 48 CHAIN
DISPLAY: PE
NO
NO
J3 48 CHARACTER IMAGE?
NO
J4 CLEAR SYSTEM STORAGE
YES
AAX100
K3 48 IMAGE WITH 120 CHAIN
DISPLAY: PF
AAX090
K4 NORMAL END OF JOB
DISPLAY: EJ

● Chart BA. System Initialization (Part 1 of 3)



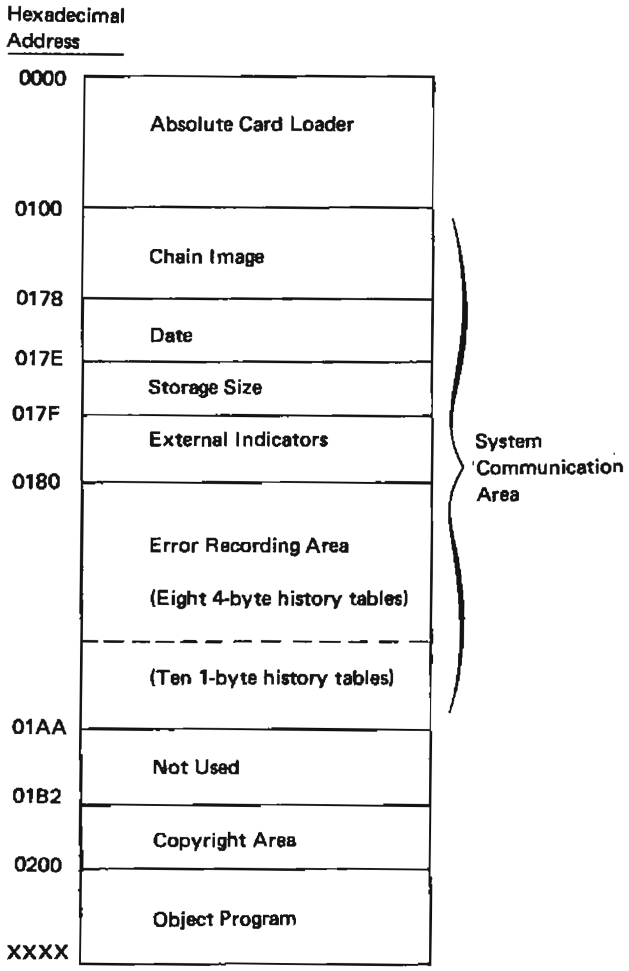
● Chart BA. System Initialization (Part 2 of 3)



● Chart BA. System Initialization (Part 3 of 3)

Section 3. Data Area Formats

The System Initialization program initializes the data areas found in the System Communication area. A storage map of the System Communication area is found in Figure 1-1.



● Figure 1-1. Storage Map for System Initialization Program

Chain Image Area - - CHAIN0

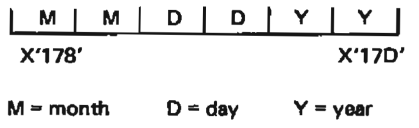
This is a 120-byte area starting at location X'100' referenced by the label CHAIN0. Each character (0-9, A-F) punched in a chain image control card is represented by four bits (the hexadecimal equivalent of the character) in the Chain Image area. Thus, two columns on the card are compressed into one byte (two hexadecimal digits) in the Chain Image area, for example:

$$\left.\begin{array}{l}\text{Card Col. 1 = A}\\ \text{Card Col. 2 = 3}\end{array}\right\}\ \text{Resulting byte = X'A3'}$$

If the printer chain image is 48 characters, the first 48 bytes of the data area are used to contain this image, and the rest of this area is set to blanks. If the image is 120 characters in length, the entire area contains the chain image.

Date Area - - DATE0

This is a 6-byte area located at location X'178' referenced by the label DATE0. This area contains the information found on the date control card in the following format:



Storage Size Area - - SIZE0

This is a 1-byte area at location X'17E' referenced by the label SIZE0. This area contains the leftmost byte of the maximum address of storage. The SIZE0 contents for the various storage sizes are as follows:

Card Entry	*SIZE0 Contents*
08	X'1F'
12	X'2F'
16	X'3F'
24	X'5F'
32	X'7F'

External Indicators - - INDO

This is a 1-byte area at location X'17F' referenced by the label INDO. This area contains eight 1-bit indicators with the following settings:

- 1 indicates switch is set on.
- 0 indicates switch is set off.

Error Recording—ENVIRO

This is a 42-byte area starting at location X'180' referenced by the label ENVIRO. This area is used as a history table of I/O device errors. The first 38 bytes of this area are punched into a card. This card is then inserted before the end card of the first module every time the System Initialization program is executed. After the card has been punched, the Error Recording area is reinitialized to hexadecimal zeros. This 42-byte area is composed of:

- Eight 4-byte areas.
  1. One Q byte from the last SIO before the error was detected.
  2. One R byte from the last SIO before the error was detected.
  3. Two sense bytes from the status sense instruction for the device in error.
- Ten 1-byte areas containing the position of the last ten line printer hammer echo checks. Each entry is converted from the value recorded by IOCS (the rightmost byte of the Line Printer Data Address Register (LPDAR) to the true print position. The results of this conversion are:

LPDAR Contents	*Error Position*	*Decimal Equivalent*
X'01'-X'7B' *(See Note)*	X'FF'	
X'94'-X'FF'	X'01'-X'6C'	01-108
X'80'-X'87'	X'6D'-X'74'	109-116
X'7C'-X'7F'	X'75'-X'78'	117-120
X'8C'-X'93'	X'79'-X'80'	121-128
X'88'-X'8B'	X'81'-X'84'	129-132

Note: An LPDAR value of X'01'-X'7B' exists only if there has been a hardware malfunction and is replaced by X'FF'.

Copyright Area

This 46-byte area starts at location X'1B2' and is initialized to blanks. The program being run fills this area with the program number and copyright information about that program.

Section 1. Introduction

The IBM System/3 Program Maintenance program is used to update decks for IBM System/3 programs when necessary. A program deck consists of all the cards which comprise an IBM System/3 program. Program decks may be comprised of smaller units known as component decks, which are separately orderable items. Component decks, in turn, may be comprised of even smaller units known as module decks. It should be noted that a single module deck may comprise a complete program deck. For example, the IBM System/3 Data Recording Program consists of one module.

The Program Maintenance program performs the following functions:

1. Replacement of component and module header cards.
2. Retention, addition, or replacement of Program Temporary Fixes (PTFs).
3. Addition or replacement of program cards in a module. A complete module may be inserted or replaced.
4. Checking of sequence, card counts, and self check information. The sequence check ensures the proper physical sequence of IBM System/3 programs. The card count ensures that the number of cards in the deck equals the number on the header card (component header card or module header card). Each card of the program deck is self checked to ensure that the card read is correctly punched.
5. Deletion of any cards which cannot be identified.
6. Printing of audit trail.

CARD FORMATS

The cards recognized by this program are the header card, PTF card, program card, and end card. (See Figure 2-1 for their format.)

There are two other cards of special interest to this program. They are the PTF header card and the End of File (/*) card. When a PTF header card is located in the Update File, the PTF Insert routine is performed. The format of this card is as follows:

P (identifies a PTF header)	cc 1
Program Number	cc 2-12
Version/Modification Level of the Component	cc 13-16
Component ID	cc 17
Unreferenced	cc 18-85
Self Check Number	cc 86-88
Unreferenced	cc 89-96

If an end-of-file (/*) card is the only card in the Update File, the Sequence Check routine is performed. The end-of-file card terminates a file. Termination of both files causes end of job. The format of this card is a /* in columns 1 and 2.

A. Header Card

Program Number	cc 20-30
Modification Level of Component	cc 31-32
Component Card Count	cc 55-58
Version of Component	cc 59-60
Module Card Count	cc 61-63
Header ID (C-Component or M-Module)	cc 64
*Update Type	cc 85
Self Check Number	cc 86-88
Module Modification Level	cc 89-90
Component ID	cc 91
Module ID	cc 92-93
Sequence Number (000)	cc 94-96

B. PTF Card

Unreferenced	cc 1-85
Self Check Number	cc 86-88
PTF ID (P)	cc 89
PTF Number	cc 90
Component ID	cc 91
Module ID	cc 92-93
Sequence Number	cc 94-96

C. Program

Unreferenced	cc 1-85
Self Check Number	cc 86-88
Module Modification Level	cc 89-90
Component ID	cc 91
Module ID	cc 92-93
Sequence Number (001-998)	cc 94-96

D. End Card

Unreferenced	cc 1-85
Self Check Number	cc 86-88
Module Modification Level	cc 89-90
Component ID	cc 91
Module ID	cc 92-93
Sequence Number (999)	cc 94-96

A. The header card is the first card of every module or component.
*The component header specifies any one of the four update types and determines:
- The update type of the first module.
- The status of all module modification levels in this component.

Subsequent module headers specify either update type 0 or update type 1, and do not determine whether or not modification levels will change.

Update types:

0=changes to a module 1=complete replacement of a module, or addition of a new module	New modification level required for a module changed or replaced.
2=changes to a module 3=complete replacement of a module, or addition of a new module	Module modification levels must remain the same.

B. PTF cards are corrections to IBM System/3 program.

C. A program card is any card that is not a PTF and has a sequence number between 001 and 998.

D. The end card is the last card of a module or component. It must have the sequence number 999.

Figure 2-1. Formats of Header, PTF, Program, and End Cards

SYSTEM REQUIREMENTS

The IBM System/3 Program Maintenance program operates using the following minimum system configuration:

- The IBM 5410 Processing Unit.
- The IBM 5424 Multi-Function Card Unit (MFCU).
- The IBM 5203 Printer.

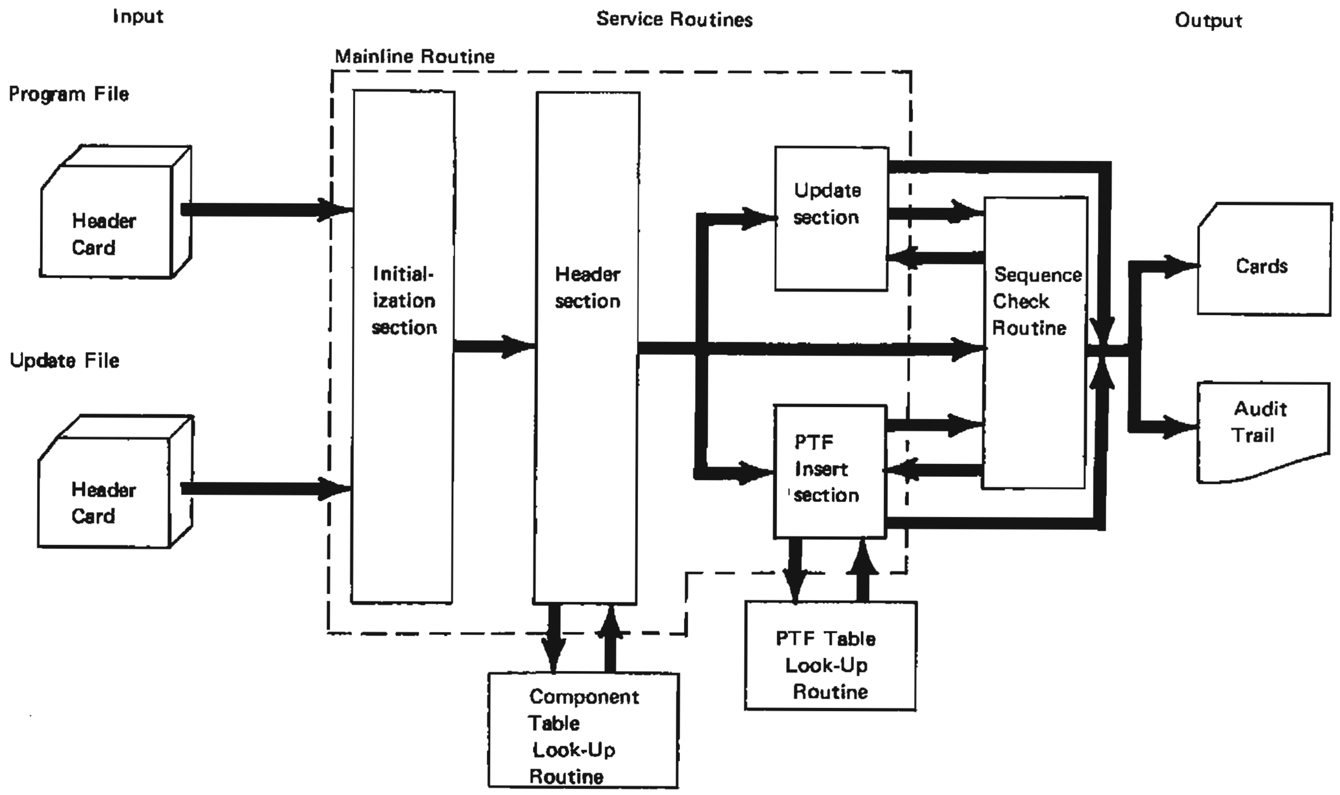
Section 2. Method of Operation

This section describes the functional flow of logic and data for the IBM System/3 Program Maintenance program (see Figures 2-2 and 2-3). The Program Organization section will describe each routine in detail.

The Program Maintenance program processes two input files, the Program File and the Update File. The Mainline routine reads the first card in each file and passes control to the Component Table Look-Up routine which builds the Component Header Table (COMPTB). When control is regained, the Header section determines the flow of logic. If the header card in the Update File was a PTF header, component header, or an end-of-file (/*) card, control is given to the PTF Insert section, Update section, or Sequence Check routine, respectively. The PTF Insert section passes control to the PTF Look-Up routine which builds a table. During execution of the Update section or the PTF Insert section, control is passed to the Sequence Check routine where the cards are sequence checked until an addition, replacement, or deletion is found. The Update section or PTF Insert section then regains control and processes the changes. When an end card is read, the Sequence Check routine reads one more card (a header card) and returns to the Mainline routine. The cards from both input files are self-checked by the Self-Check routine.

The Update File and Program File are read into two separate input buffers (UPDATE and PRGRD). After the header cards are read by the Mainline routine, the Component Header Table (COMPTB) is built from the header card in the Program File. The Update File deck is then read by the Sequence Check routine until the first sequence break or PTF is reached. The Program File deck is then deleted up to this point by the Update section (on an Update run). The sequence break is filled in by the Sequence Check routine reading from the Program File until a card is found with a sequence number equal to or greater than the sequence number on the card in the Update File. The Update section or the PTF Insert section then handles the card in the Update File or in the Program File as appropriate. The cycle is repeated until the end of the module is reached on both the Program File and the Update File.

If a complete module replacement is to be processed, the preceding procedure is bypassed and the Sequence Check



NOTE: All Program Maintenance routines call the Self-Check routine.

Figure 2-2. Control Flow of Program Maintenance Program

routine reads and checks the cards. For a PTF Insert run, PTF header information is stored in a PTF Table (PTFTAB). PTFHD is the hold area for this data.

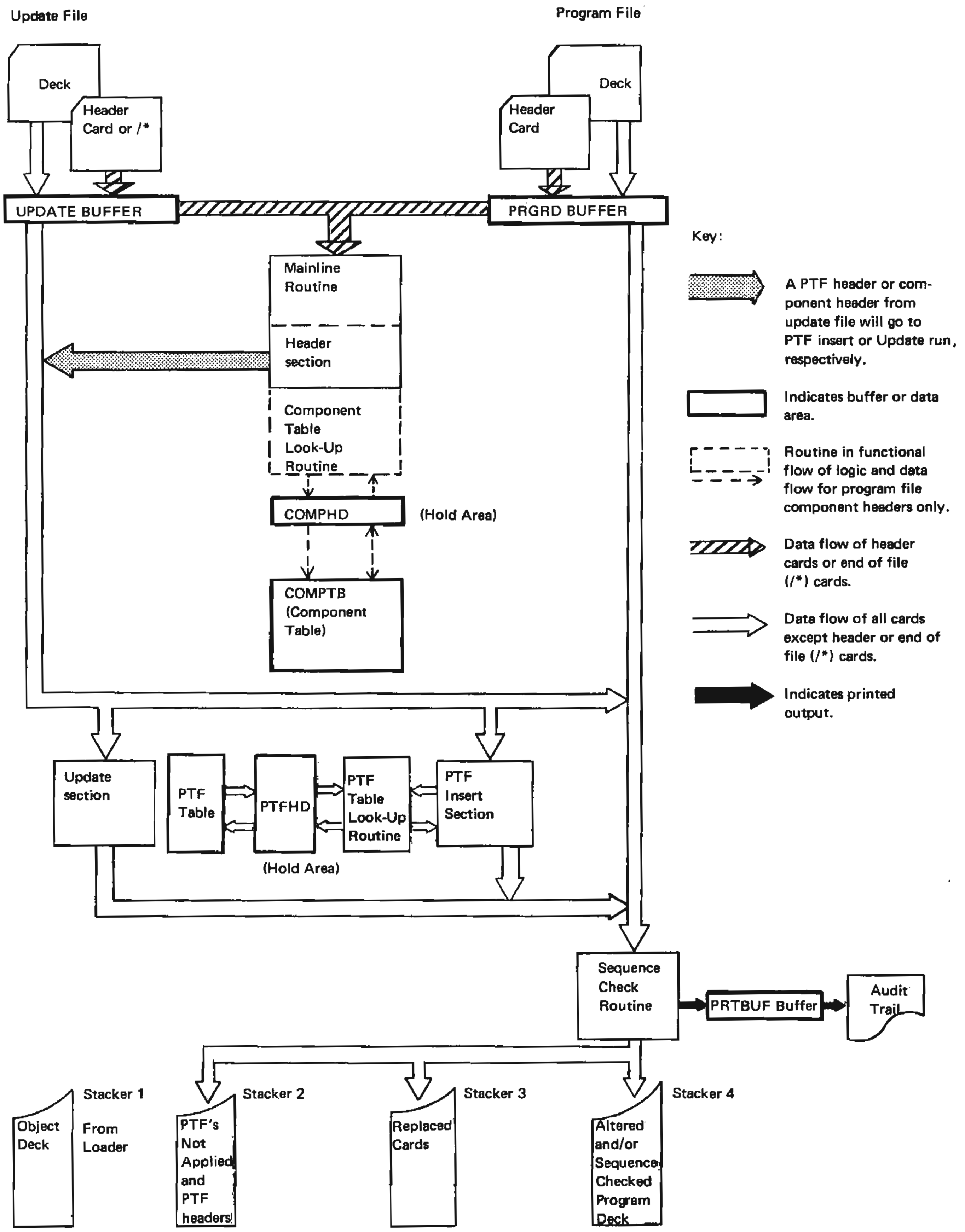
For each type of run an audit trail is printed (PRTBUF is the print buffer), and the processed cards are routed to appropriate stackers as follows:

- Update run
  Stacker 1—Program Maintenance program object deck.
  Stacker 2—PTFs not applied.
  Stacker 3—Cards replaced by Update.
  Stacker 4—Altered program deck.

- PTF Insert run
  Stacker 1—Program Maintenance program object deck.
  Stacker 2—PTF headers.
  Stacker 3—Cards replaced by PTFs.
  Stacker 4—Altered program deck.

- Sequence Check run
  Stacker 1—Program Maintenance program object deck.
  Stacker 2—Not used.
  Stacker 3—Not used.
  Stacker 4—Sequence-checked deck.

- EB Insert run
  Stacker 1—Program Maintenance program object deck.
  Stacker 2—Not used.
  Stacker 3—Not used.
  Stacker 4—RPG II Compiler deck containing the EB phases.



Update File
Program File
Deck
Header Card or /*
Deck
Header Card
UPDATE BUFFER
PRGRD BUFFER
Mainline Routine
Header section
Component Table Look-Up Routine
COMPHD
(Hold Area)
COMPTB (Component Table)
Update section
PTF Table
PTFHD
PTF Table Look-Up Routine
PTF Insert Section
(Hold Area)
Sequence Check Routine
PRTBUF Buffer
Audit Trail
Stacker 1
Object Deck
From Loader
Stacker 2
PTF's Not Applied and PTF headers
Stacker 3
Replaced Cards
Stacker 4
Altered and/or Sequence Checked Program Deck
Key:
A PTF header or component header from update file will go to PTF insert or Update run, respectively.
Indicates buffer or data area.
Routine in functional flow of logic and data flow for program file component headers only.
Data flow of header cards or end of file (/*) cards.
Data flow of all cards except header or end of file (/*) cards.
Indicates printed output.

Figure 2-3. Functional Flow of Data for Program Maintenance Program

Section 3. Program Organization

This section shows how the routines that comprise the Program Maintenance program are interconnected and the function of each routine. Figure 2-4 is a storage map for the program.

Hex Address	
0000	Update File Read Buffer
007C	Print Buffer
0100	System Communication Area
01B2	Constants and Data Areas
0280	Program File Read Buffer
02E0	Component Header Table PTF Header Table PTF Identification Table Printer Messages IOCS Constants
0598	Mainline Routine Initialization section Header section Update section PTF Insert section Self-Check Routine Sequence Check Routine Component Table Look-Up Routine PTF Table Look-Up Routine Single Carriage Line Printer IOCS Read Both Hoppers MFCU IOCS
17FC	Unused
XXXX	

Figure 2-4. Storage Map for Program Maintenance Program

General Overview of the Program Maintenance Program

The functions of this program are to sequence check, insert PTF's or insert Updates. The first card in the Update File determines which routine will be run.

1. An end-of-file (/*) card indicates that the Sequence Check routine is to be run.
2. A P in column 1 identifies a PTF header card which indicates the PTF Insert section is to be run. Every component to be updated by a PTF has a PTF header card.
3. If there is a C in column 64 and 000 in columns 94-96 (Sequence Number) of the first card in the Update File, the Update section is run.
4. If there is an M in column 64 and GEB000 in columns 91-96 of the first card in the Update File, the EB Insert run is processed.

For information on the IOCS routines used, refer to the *IBM System/3 Card System Absolute Card Loader and Input/Output Control System Logic Manual,* SY21-0521.

Mainline Routine

Initialization section (APMAA1)

Chart: EA (part 1 and 2)

Functions:

- Initializes registers.
- Initializes data area information and switches.
- Reads the first card from the Program File and the Update File.
- Prints component deck total information at the end of each program deck.

Subroutines Used:'

- Self-Check Routine
- Read Only - Both Hoppers (Object) MFCU IOCS
- Line Printer - Single Feed Carriage (Object) Printer IOCS

Header section (AAB010)

Chart: EA (part 3)

Functions:

- Checks component header information for validity.

    1. Check to determine if the module is in the current program. If it is not:
        a. If the header is a component header, a new program deck is being updated and the functional flow of the program returns to the Mainline routine.
        b. If the header is not a component header, a halt occurs.
    2. Checks the module ID for proper sequence. Each module ID must be greater than the previous module ID, except in the RPG Compiler where several EB phases (same module ID) are allowed. If a module is out of sequence, a halt occurs and control returns to the Mainline routine.
    3. Checks the header card sequence number against 000. If they do not agree, a halt occurs.

- Places component header information into the Component Header Table and into the Component Table Hold Area (COMPHD) via the Component Table Look-Up routine.

- Saves data from the header card for the Sequence Check routine.

Subroutines Used:

- Component Table Look-Up Routine

- Sequence Check Routine

- Self-Check Routine

Update section (AAC010)

Chart: EA (parts 4, 5, 6, 7, 8, 9, 10, 11, 12, and 13)

Functions:

- Checks the validity of the header card.

    1. Checks the sequence number against 000. If they do not agree, control is given to the PTF Insert routine.
    2. Checks for an M or C in column 64. If there is none, control is given to the PTF Insert section.
    3. Checks to determine whether this header card is for the current program. If not, a halt occurs.
    4. Checks the module ID for proper sequence. Each module ID must be greater than the previous module ID, except in the RPG Compiler where several EB phases (same module ID) are allowed. If a module is out of sequence, a halt occurs and control returns to the Mainline routine.
    5. Checks first header card for EB module header. If it is, a switch is set for EB Insert run.

- Replaces old header cards with new header cards.

- Inserts Updates:

    1. Within the body of a module, replacing existing cards.
    2. Complete module insertion.
    3. Complete module replacement.
    4. As replacement for PTFs outside the body of a module.
    5. As addition to the body of a module.

- Updates IOB to allow Sequence Check routine to read from the Update File for a complete module replacement.

- Prints all PTFs in the module.

- Prints deleted PTFs.

Subroutines Used:

- Read Only - Both Hoppers (Object) MFCU IOCS

- Line Printer - Single Feed Carriage (Object) Printer IOCS

- Sequence Check Routine

Program Temporary Fix (PTF) Insert section (AAE010)

Chart: EA (parts 14, 15, and 16)

Functions:

- Checks the PTF Header card for a P in column 1. If there is none, a halt occurs.
- Places PTF header information in the PTF Header Table via the PTF Table Look-Up routine.
- Inserts PTFs.

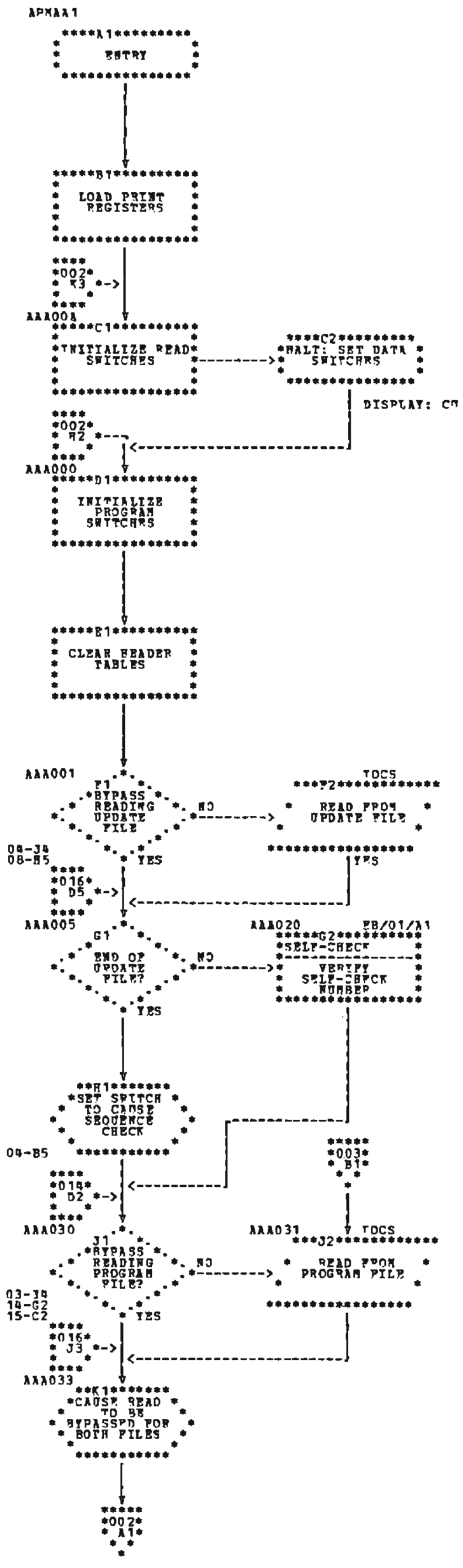
  1. Inside the body of a program deck on a one for one replacement basis.
  2. Outside the body of a program deck. This may be a one for one replacement or it may be a complete insertion with nothing deleted.

- Prints PTF IDs and sequence IDs of all PTFs in the module, including the inserted PTFs.
- Prints deleted PTFs.

Subroutines Used:

- Read Only - Both Hoppers (Object) MFCU IOCS
- Line Printer - Single Feed Carriage (Object) Printer IOCS
- PTF Look-Up Routine
- Sequence Check Routine
- Self-Check Routine

Initializaton Section



● Chart EA. Mainline Routine (Part 1 of 16)

Initialization Section

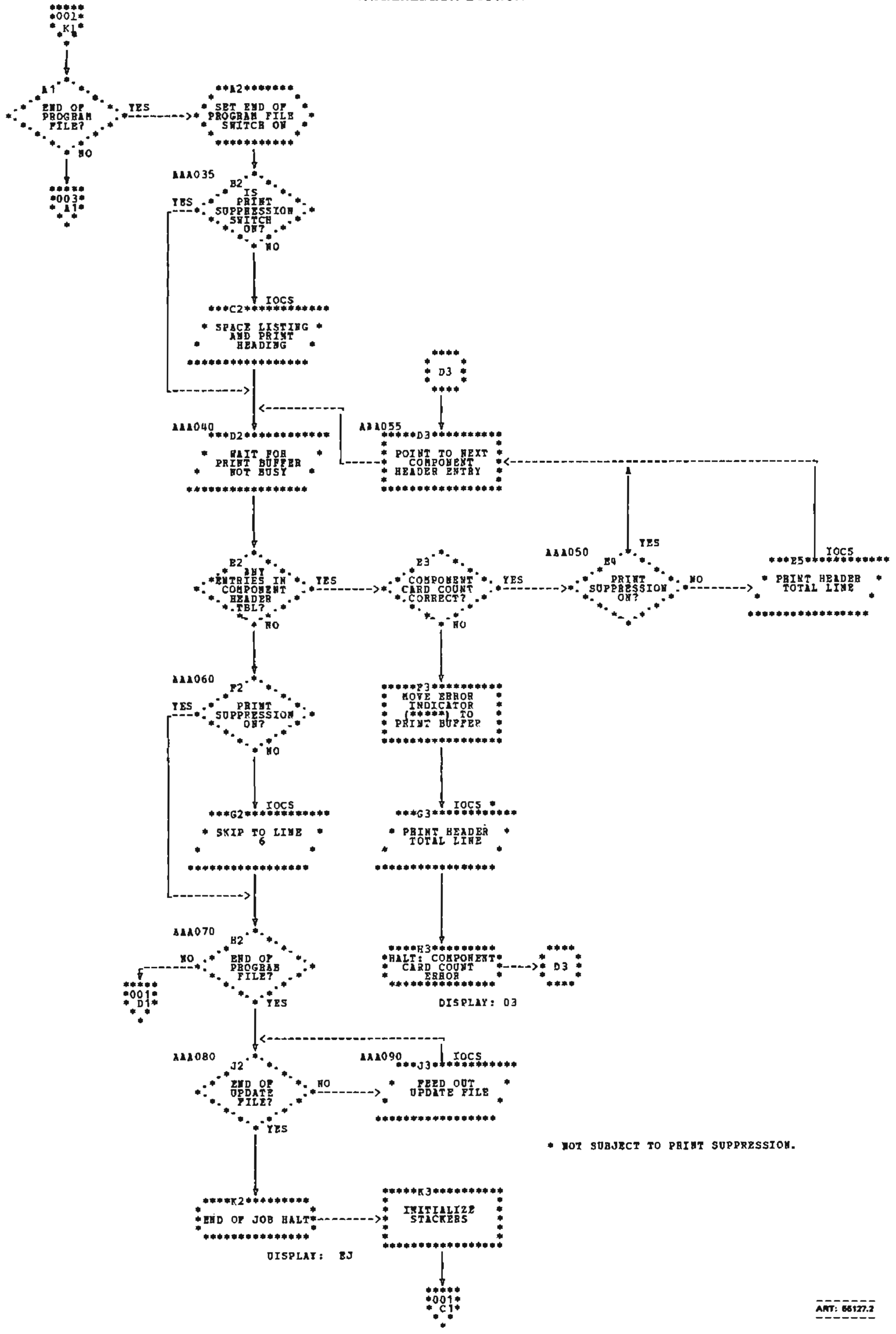

001 K1
A1 END OF PROGRAM FILE?
YES
NO
003 A1
A2 SET END OF PROGRAM FILE SWITCH ON
AAA035 B2 IS PRINT SUPPRESSION SWITCH ON?
YES
NO
IOCS C2 SPACE LISTING AND PRINT HEADING
AAA040 D2 WAIT FOR PRINT BUFFER NOT BUSY
D3
AAA055 D3 POINT TO NEXT COMPONENT HEADER ENTRY
E2 ANY ENTRIES IN COMPONENT HEADER TBL?
YES
NO
E3 COMPONENT CARD COUNT CORRECT?
YES
NO
AAA050 E4 PRINT SUPPRESSION ON?
YES
NO
IOCS E5 PRINT HEADER TOTAL LINE
AAA060 F2 PRINT SUPPRESSION ON?
YES
NO
F3 MOVE ERROR INDICATOR (*****) TO PRINT BUFFER
IOCS G2 SKIP TO LINE 6
IOCS * G3 PRINT HEADER TOTAL LINE
AAA070 H2 END OF PROGRAM FILE?
NO
YES
001 D1
H3 HALT: COMPONENT CARD COUNT ERROR
D3
DISPLAY: 03
AAA080 J2 END OF UPDATE FILE?
NO
YES
AAA090 IOCS J3 FEED OUT UPDATE FILE
* NOT SUBJECT TO PRINT SUPPRESSION.
K2 END OF JOB HALT
DISPLAY: EJ
K3 INITIALIZE STACKERS
001 C1
ART: 66127.2

Chart EA. Mainline Routine (Part 2 of 16)

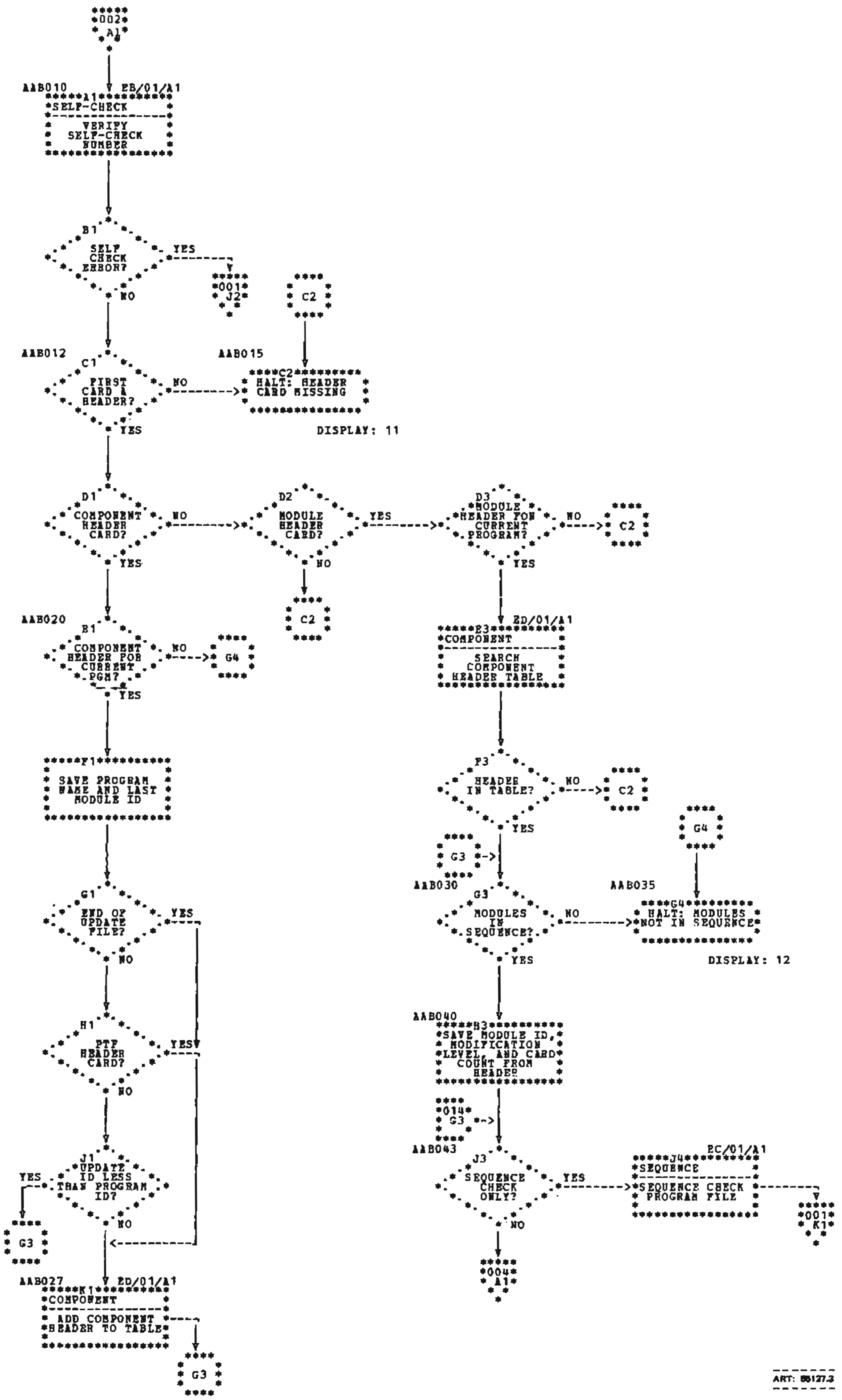
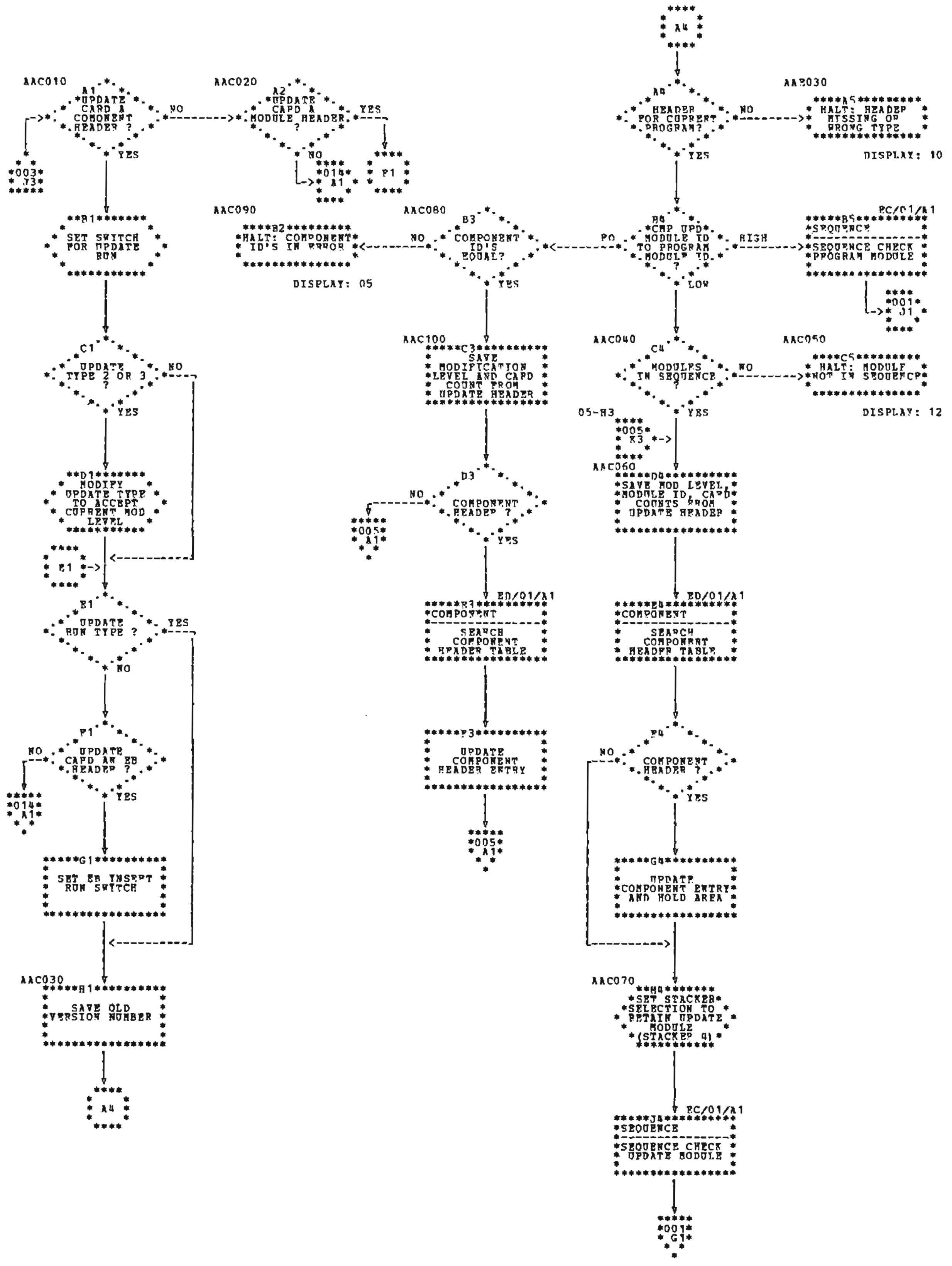
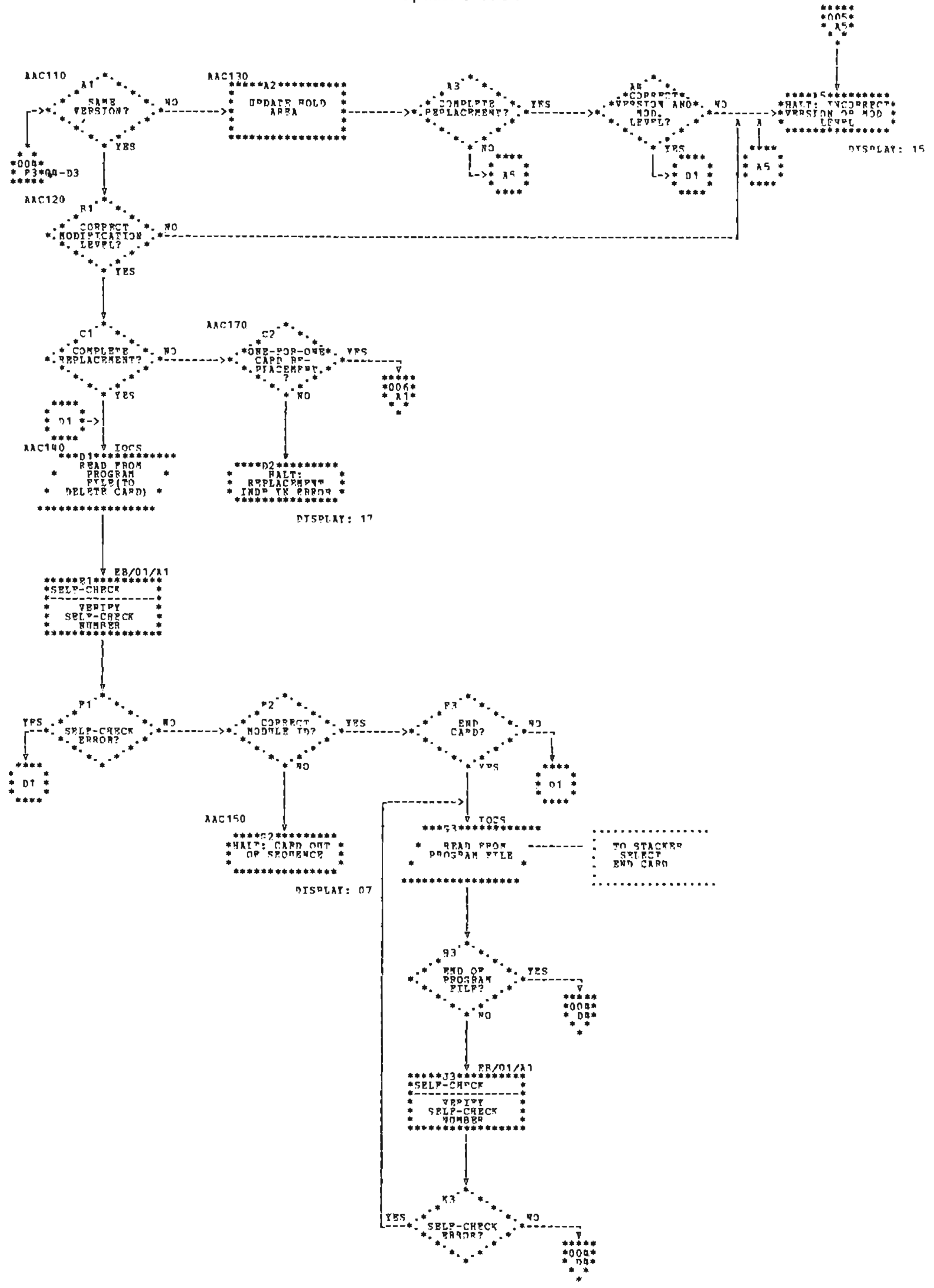


Chart EA. Mainline Routine (Part 3 of 16)


A4
AAC010
A1 UPDATE CARD A COMONENT HEADER ?
NO
YES
003 J3
AAC020
A2 UPDATE CARD A MODULE HEADER ?
YES
NO
014 A1
P1
A4 HEADER FOR CURRENT PROGRAM?
NO
YES
AA3030
A5 HALT: HEADER MISSING OR WRONG TYPE
DISPLAY: 10
B1 SET SWITCH FOR UPDATE RUN
AAC090
B2 HALT: COMPONENT ID'S IN ERROR
DISPLAY: 05
AAC080
B3 COMPONENT ID'S EQUAL?
NO
YES
B4 CMP UPD MODULE ID TO PROGRAM MODULE ID ?
HIGH
LOW
EC/01/A1
B5 SEQUENCE
SEQUENCE CHECK PROGRAM MODULE
001 J1
C1 UPDATE TYPE 2 OR 3 ?
NO
YES
AAC100
C3 SAVE MODIFICATION LEVEL AND CARD COUNT FROM UPDATE HEADER
AAC040
C4 MODULES IN SEQUENCE ?
NO
YES
AAC050
C5 HALT: MODULE NOT IN SEQUENCE
DISPLAY: 12
05-H3
005 K3
D1 MODIFY UPDATE TYPE TO ACCEPT CURRENT MOD LEVEL
D3 COMPONENT HEADER ?
NO
YES
005 A1
AAC060
D4 SAVE MOD LEVEL, MODULE ID, CARD COUNTS FROM UPDATE HEADER
E1
E1 UPDATE RUN TYPE ?
YES
NO
ED/01/A1
E3 COMPONENT
SEARCH COMPONENT HEADER TABLE
ED/01/A1
E4 COMPONENT
SEARCH COMPONENT HEADER TABLE
F1 UPDATE CARD AN EB HEADER ?
NO
YES
014 A1
F3 UPDATE COMPONENT HEADER ENTRY
F4 COMPONENT HEADER ?
NO
YES
005 A1
G1 SET EB INSERT RUN SWITCH
G4 UPDATE COMPONENT ENTRY AND HOLD AREA
AAC030
H1 SAVE OLD VERSION NUMBER
AAC070
H4 SET STACKER SELECTION TO RETAIN UPDATE MODULE (STACKER 4)
A4
EC/01/A1
J4 SEQUENCE
SEQUENCE CHECK UPDATE MODULE
001 G1

● Chart EA. Mainline Routine (Part 4 of 16)



● Chart EA. Mainline Routine (Part 5 of 16)

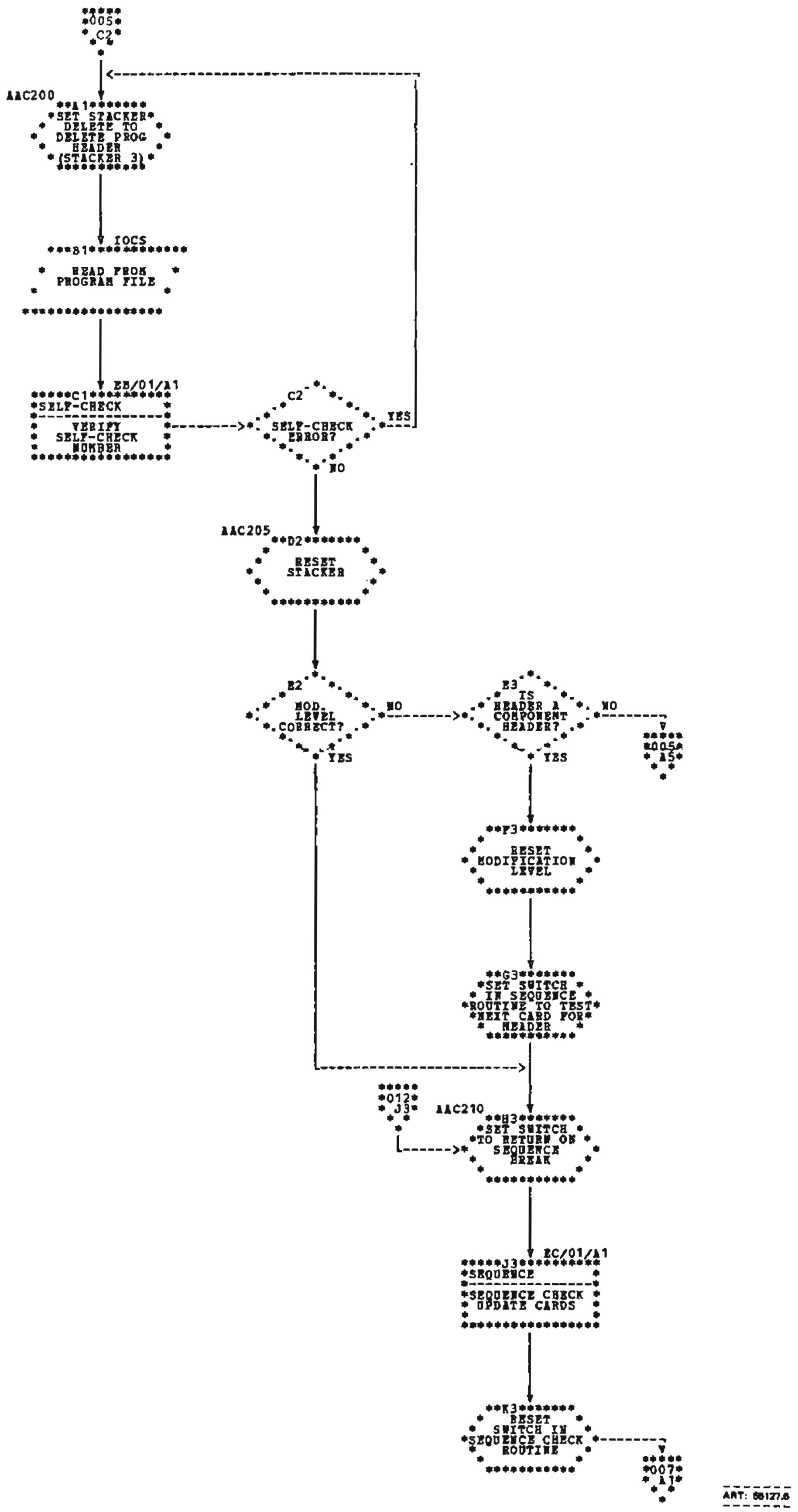

005 C2
AAC200
A1 SET STACKER DELETE TO DELETE PROG HEADER (STACKER 3)
IOCS
B1 READ FROM PROGRAM FILE
EB/01/A1
C1 SELF-CHECK
VERIFY SELF-CHECK NUMBER
C2 SELF-CHECK ERROR?
YES
NO
AAC205
D2 RESET STACKER
E2 MOD. LEVEL CORRECT?
NO
YES
E3 IS HEADER A COMPONENT HEADER?
NO
YES
005 A5
F3 RESET MODIFICATION LEVEL
G3 SET SWITCH IN SEQUENCE ROUTINE TO TEST NEXT CARD FOR HEADER
012 J3
AAC210
H3 SET SWITCH TO RETURN ON SEQUENCE BREAK
EC/01/A1
J3 SEQUENCE
SEQUENCE CHECK UPDATE CARDS
K3 RESET SWITCH IN SEQUENCE CHECK ROUTINE
007 A1
ART: 66127.6

Chart EA. Mainline Routine (Part 6 of 16)

Update Section

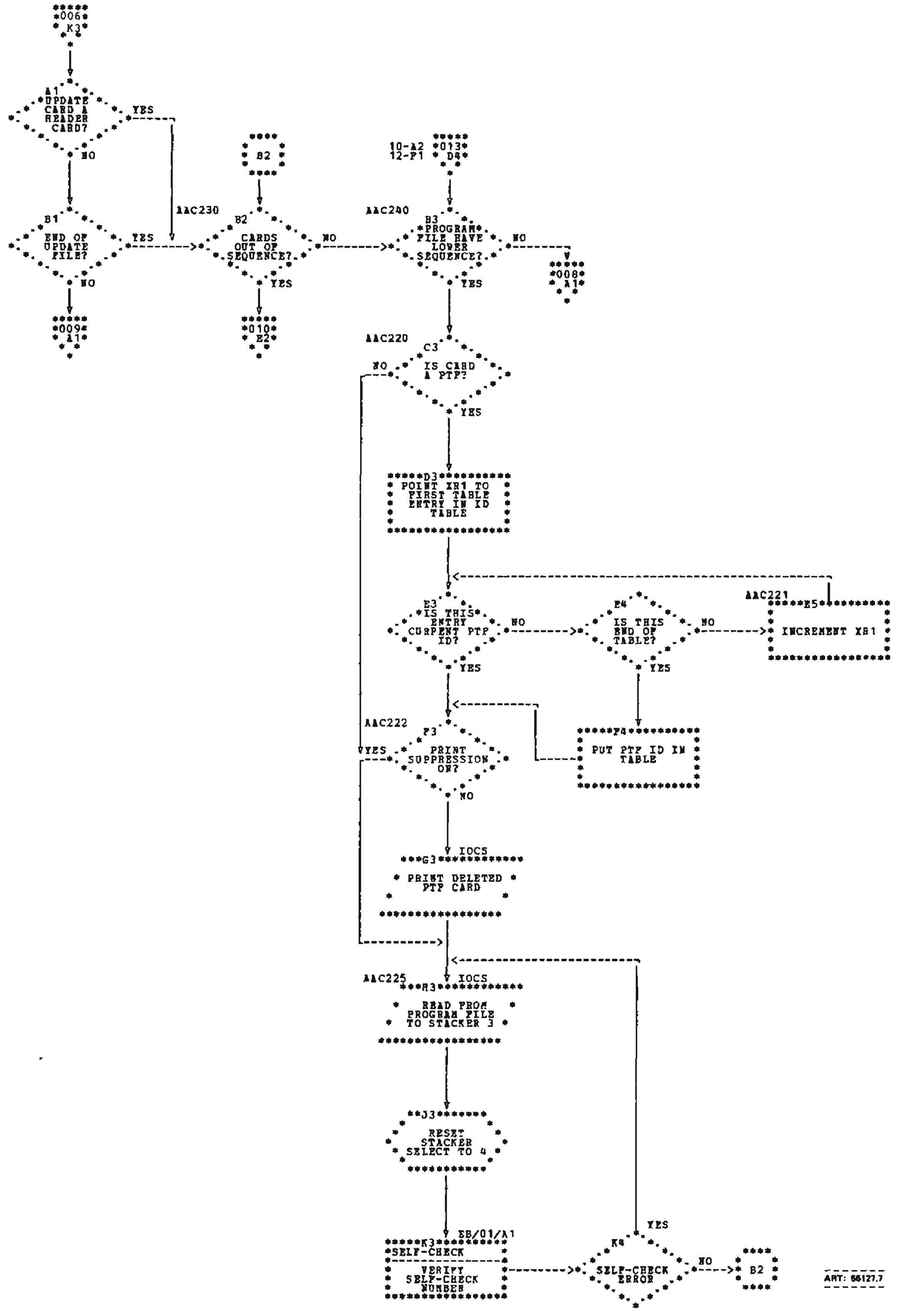


Chart EA. Mainline Routine (Part 7 of 16)

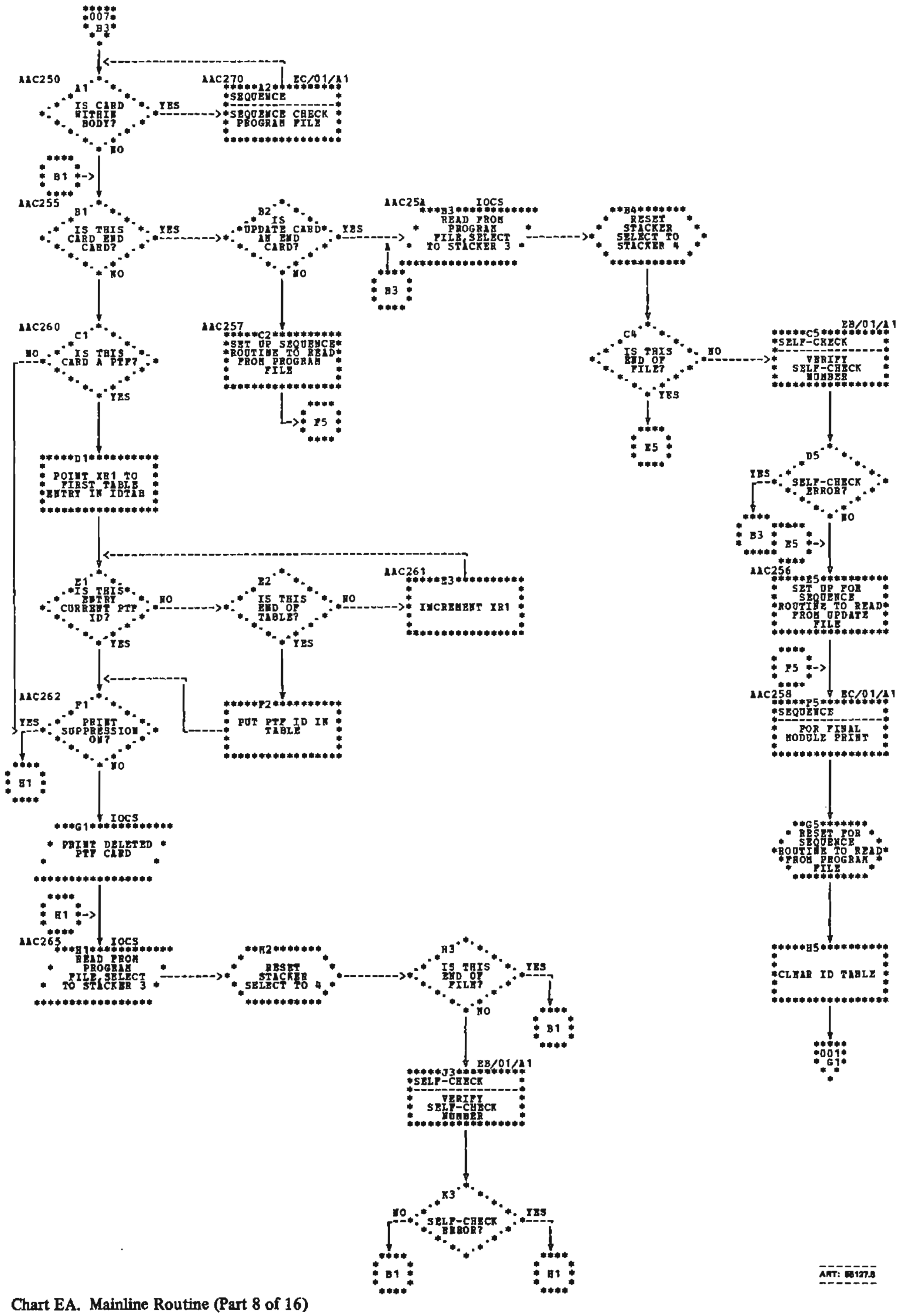

007 B3
AAC250 A1 IS CARD WITHIN BODY?
YES
NO
AAC270 A2 EC/01/A1 SEQUENCE SEQUENCE CHECK PROGRAM FILE
B1
AAC255 B1 IS THIS CARD END CARD?
YES
NO
B2 IS UPDATE CARD AN END CARD?
YES
NO
AAC25A B3 IOCS READ FROM PROGRAM FILE,SELECT TO STACKER 3
B3
B4 RESET STACKER SELECT TO STACKER 4
AAC260 C1 IS THIS CARD A PTF?
NO
YES
AAC257 C2 SET UP SEQUENCE ROUTINE TO READ FROM PROGRAM FILE
F5
C4 IS THIS END OF FILE?
NO
YES
E5
EB/01/A1 C5 SELF-CHECK VERIFY SELF-CHECK NUMBER
D5 SELF-CHECK ERROR?
YES
NO
B3
E5
D1 POINT XR1 TO FIRST TABLE ENTRY IN IDTAB
E1 IS THIS ENTRY CURRENT PTF ID?
NO
YES
E2 IS THIS END OF TABLE?
NO
YES
AAC261 E3 INCREMENT XR1
AAC256 E5 SET UP FOR SEQUENCE ROUTINE TO READ FROM UPDATE FILE
F5
AAC258 F5 EC/01/A1 SEQUENCE FOR FINAL MODULE PRINT
AAC262 F1 PRINT SUPPRESSION ON?
YES
NO
H1
F2 PUT PTF ID IN TABLE
G1 IOCS PRINT DELETED PTF CARD
G5 RESET FOR SEQUENCE ROUTINE TO READ FROM PROGRAM FILE
H1
AAC265 H1 IOCS READ FROM PROGRAM FILE,SELECT TO STACKER 3
H2 RESET STACKER SELECT TO 4
H3 IS THIS END OF FILE?
YES
NO
B1
H5 CLEAR ID TABLE
001 G1
EB/01/A1 J3 SELF-CHECK VERIFY SELF-CHECK NUMBER
K3 SELF-CHECK ERROR?
NO
YES
B1
H1
ART: 88127.8

Chart EA. Mainline Routine (Part 8 of 16)

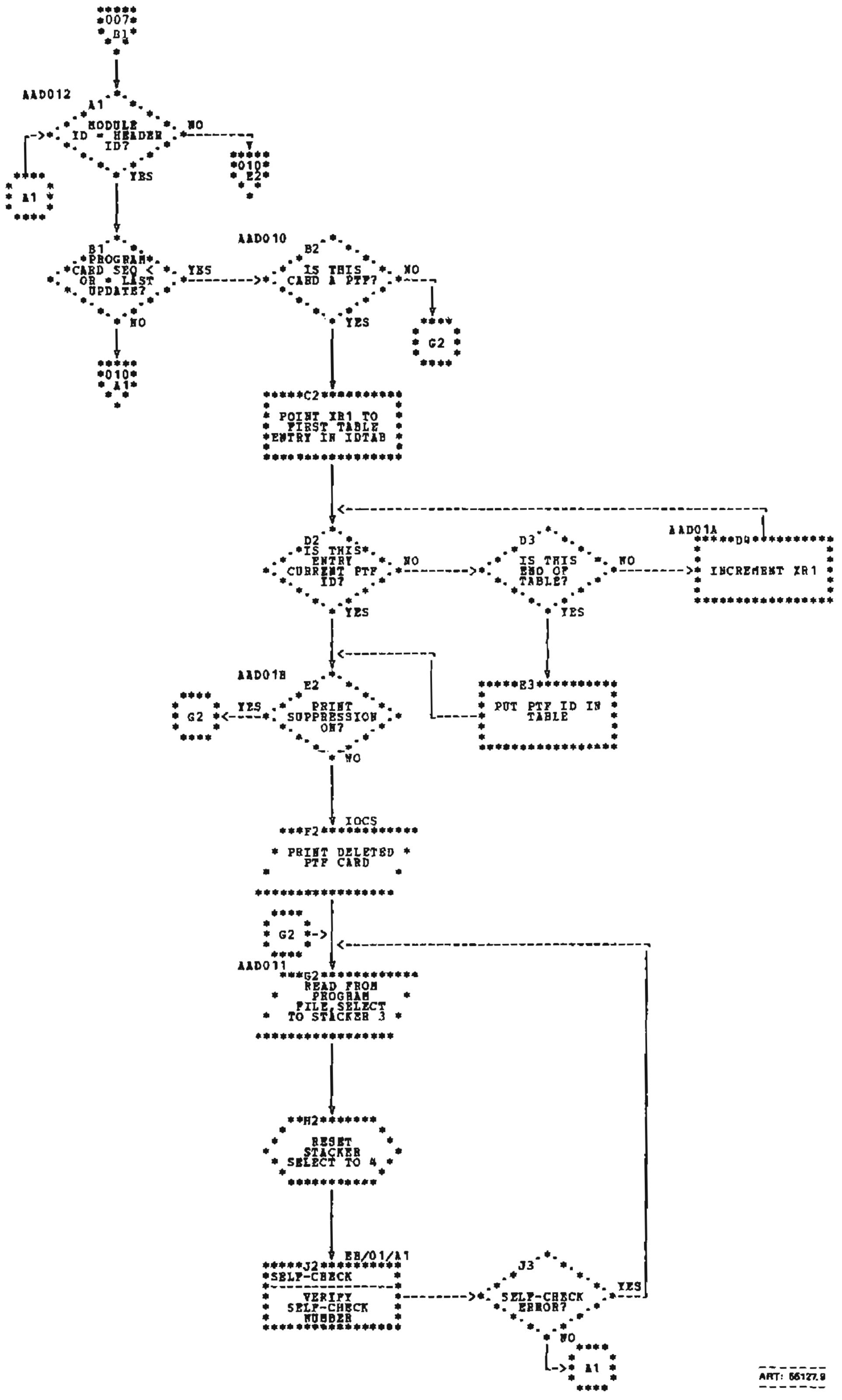
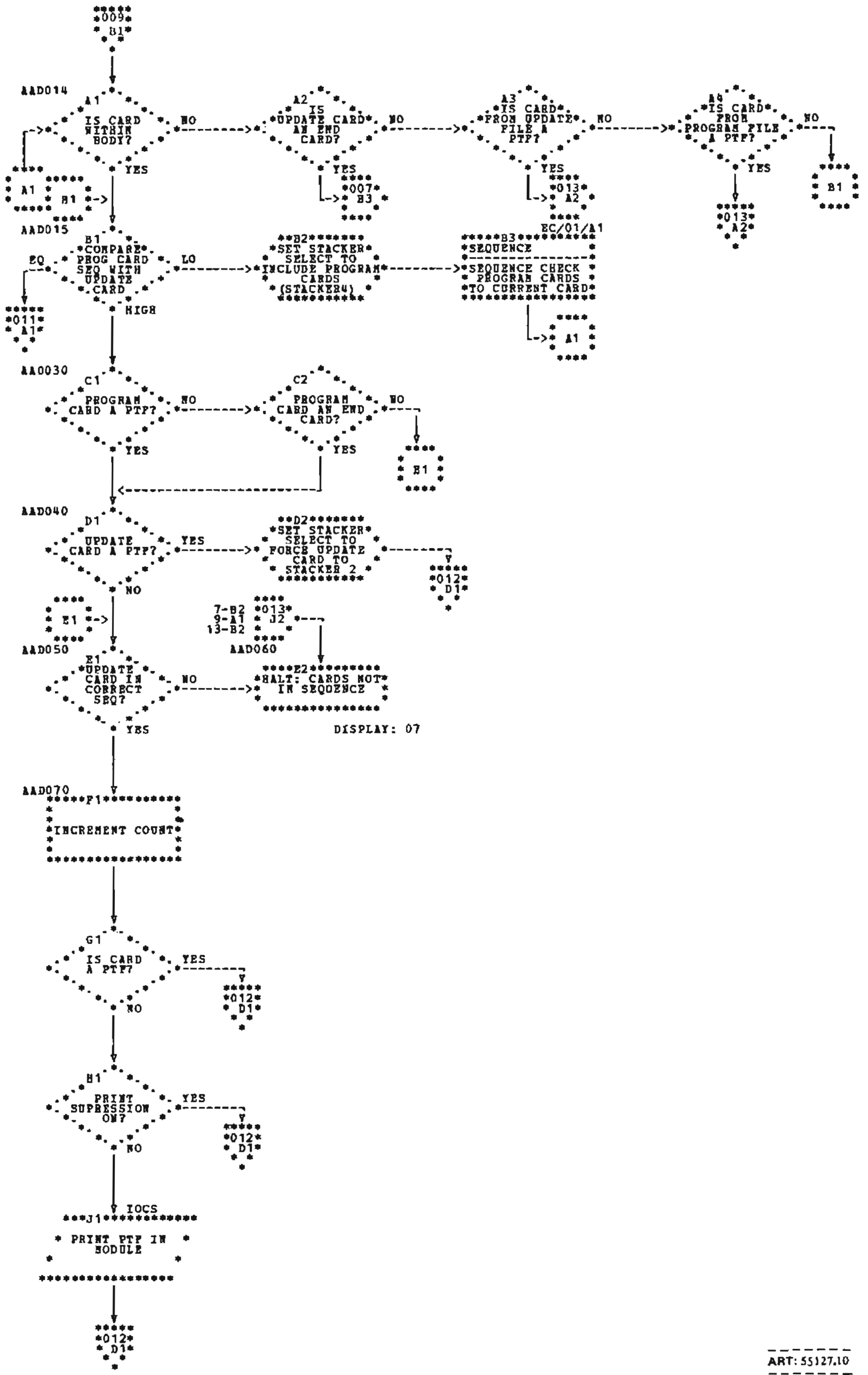


Chart EA. Mainline Routine (Part 9 of 16)

Update Section


009 B1
AAD014
A1 IS CARD WITHIN BODY?
NO
YES
A1
B1
A2 IS UPDATE CARD AN END CARD?
NO
YES
007 B3
A3 IS CARD FROM UPDATE FILE A PTF?
NO
YES
013 A2
A4 IS CARD FROM PROGRAM FILE A PTF?
NO
YES
013 A2
B1
AAD015
B1 COMPARE PROG CARD SEQ WITH UPDATE CARD
EQ
LO
HIGH
011 A1
B2 SET STACKER SELECT TO INCLUDE PROGRAM CARDS (STACKER4)
EC/01/A1
B3 SEQUENCE
SEQUENCE CHECK PROGRAM CARDS TO CURRENT CARD
A1
AA0030
C1 PROGRAM CARD A PTF?
NO
YES
C2 PROGRAM CARD AN END CARD?
NO
YES
B1
AAD040
D1 UPDATE CARD A PTF?
YES
NO
D2 SET STACKER SELECT TO FORCE UPDATE CARD TO STACKER 2
012 D1
E1
7-B2
9-A1
13-B2
013 J2
AAD060
AAD050
E1 UPDATE CARD IN CORRECT SEQ?
NO
YES
E2 HALT: CARDS NOT IN SEQUENCE
DISPLAY: 07
AAD070
F1 INCREMENT COUNT
G1 IS CARD A PTF?
YES
NO
012 D1
H1 PRINT SUPRESSION ON?
YES
NO
012 D1
IOCS
J1 PRINT PTF IN MODULE
012 D1

ART: 55127.10

Chart EA. Mainline Routine (Part 10 of 16)

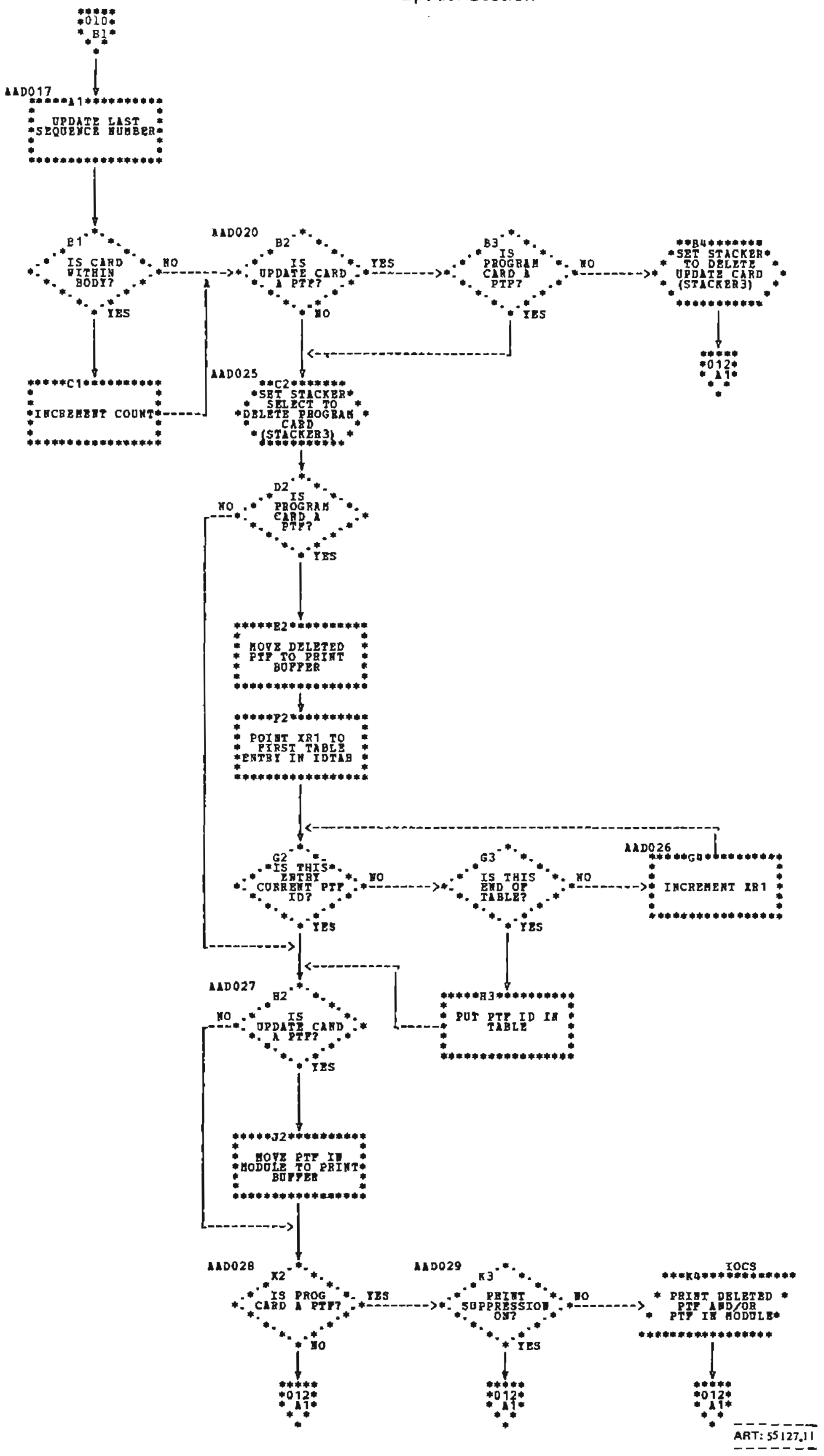
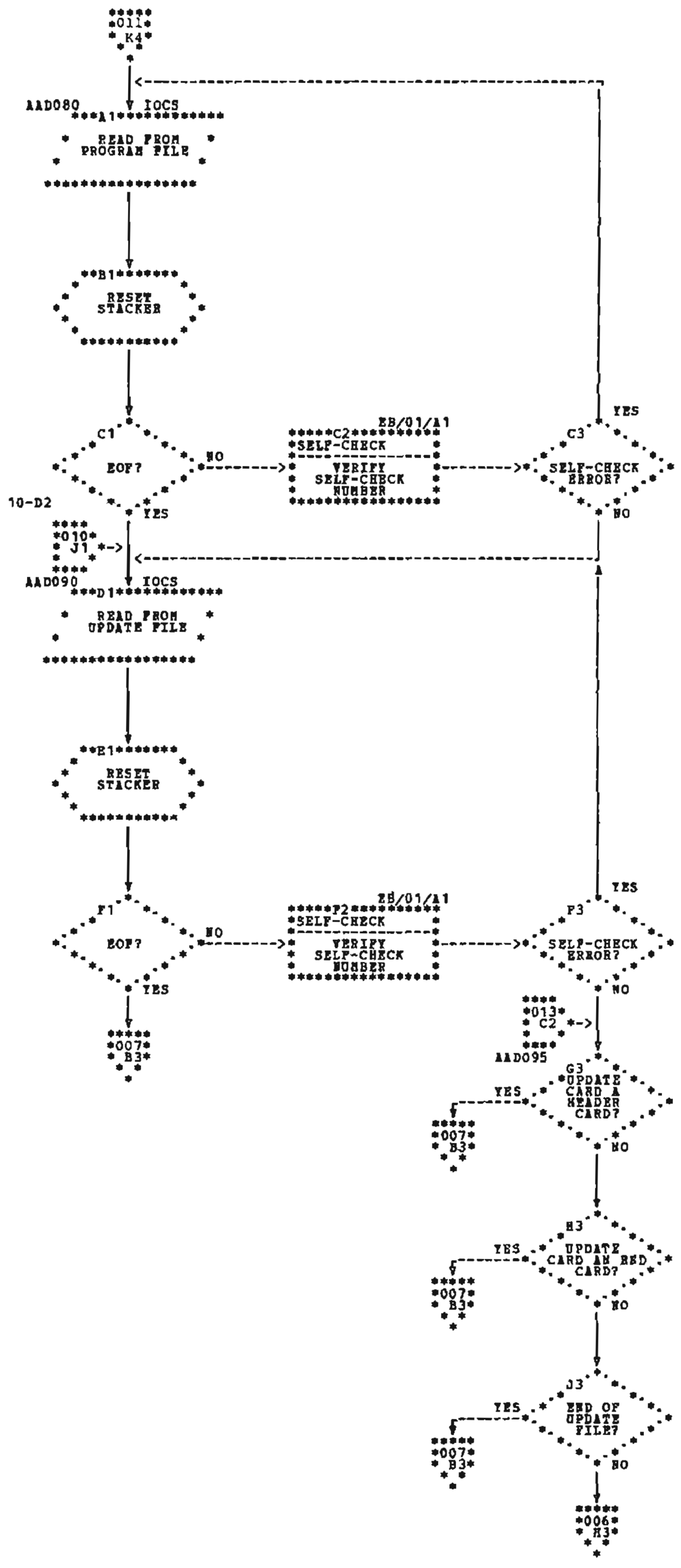

010 B1
AAD017 A1 UPDATE LAST SEQUENCE NUMBER
B1 IS CARD WITHIN BODY?
NO
YES
C1 INCREMENT COUNT
AAD020 B2 IS UPDATE CARD A PTF?
YES
NO
B3 IS PROGRAM CARD A PTF?
NO
YES
B4 SET STACKER TO DELETE UPDATE CARD (STACKER3)
012 A1
AAD025 C2 SET STACKER SELECT TO DELETE PROGRAM CARD (STACKER3)
D2 IS PROGRAM CARD A PTF?
NO
YES
E2 MOVE DELETED PTF TO PRINT BUFFER
F2 POINT XR1 TO FIRST TABLE ENTRY IN IDTAB
G2 IS THIS ENTRY CURRENT PTF ID?
NO
YES
G3 IS THIS END OF TABLE?
NO
YES
AAD026 G4 INCREMENT XR1
H3 PUT PTF ID IN TABLE
AAD027 H2 IS UPDATE CARD A PTF?
NO
YES
J2 MOVE PTF IN MODULE TO PRINT BUFFER
AAD028 K2 IS PROG CARD A PTF?
YES
NO
AAD029 K3 PRINT SUPPRESSION ON?
NO
YES
IOCS K4 PRINT DELETED PTF AND/OR PTF IN MODULE
012 A1
012 A1
012 A1
ART: 55127.11

Chart EA. Mainline Routine (Part 11 of 16)

Update Section



ART: 55127.12

Chart EA. Mainline Routine (Part 12 of 16)

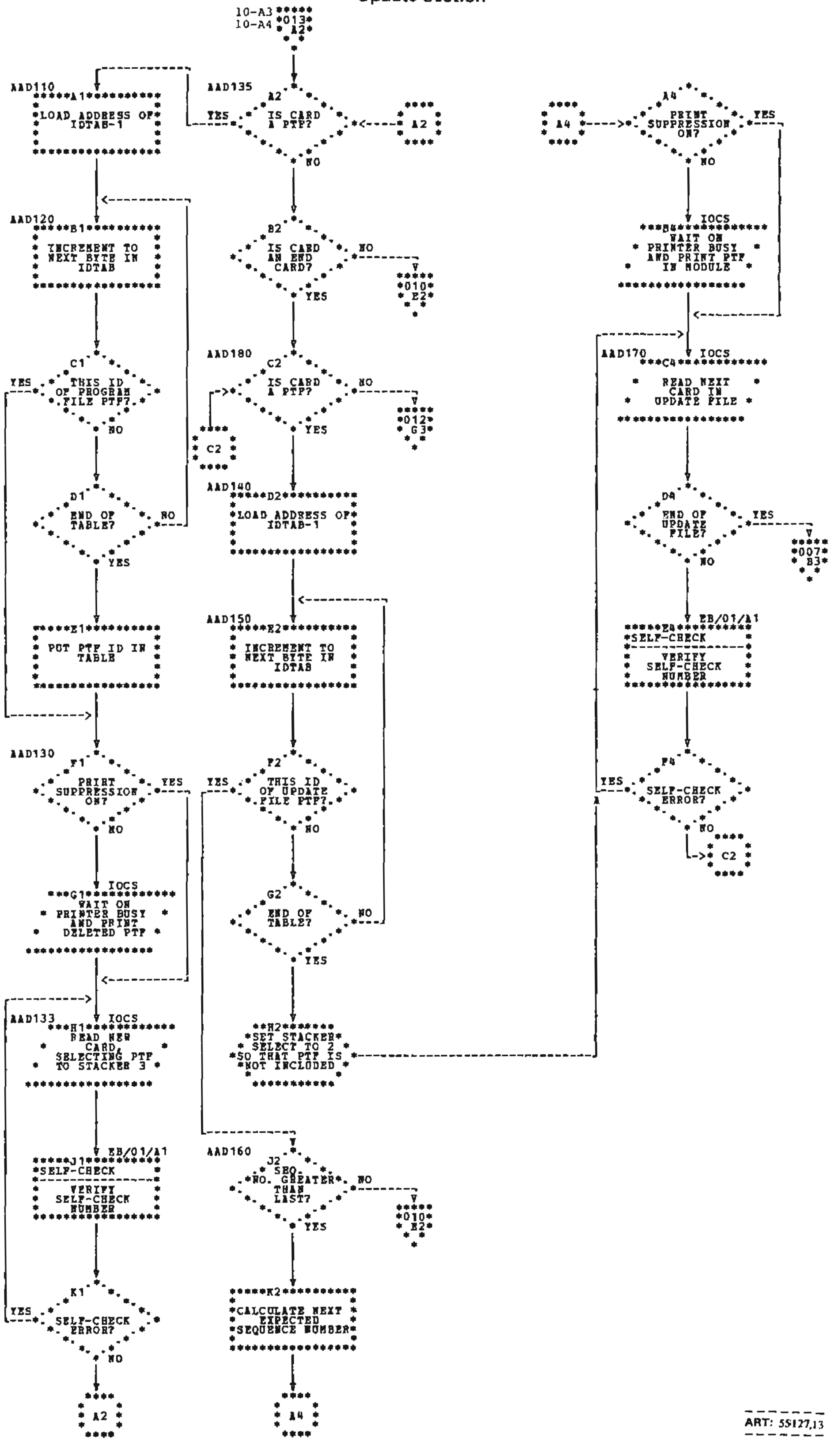
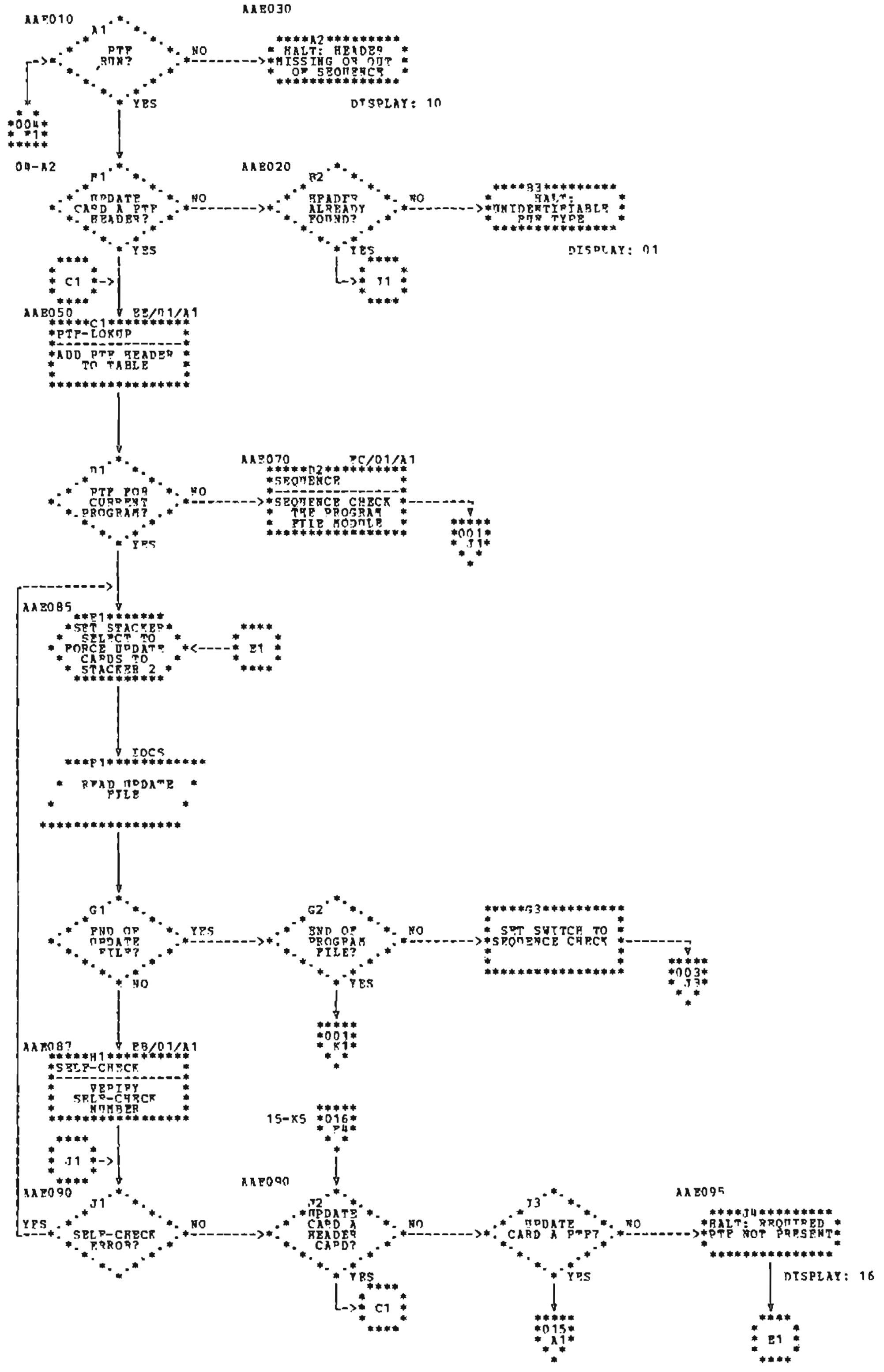


Chart EA. Mainline Routine (Part 13 of 16)

PTF Insert Section



● Chart EA. Mainline Routine (Part 14 of 16)

PTF Insert Section

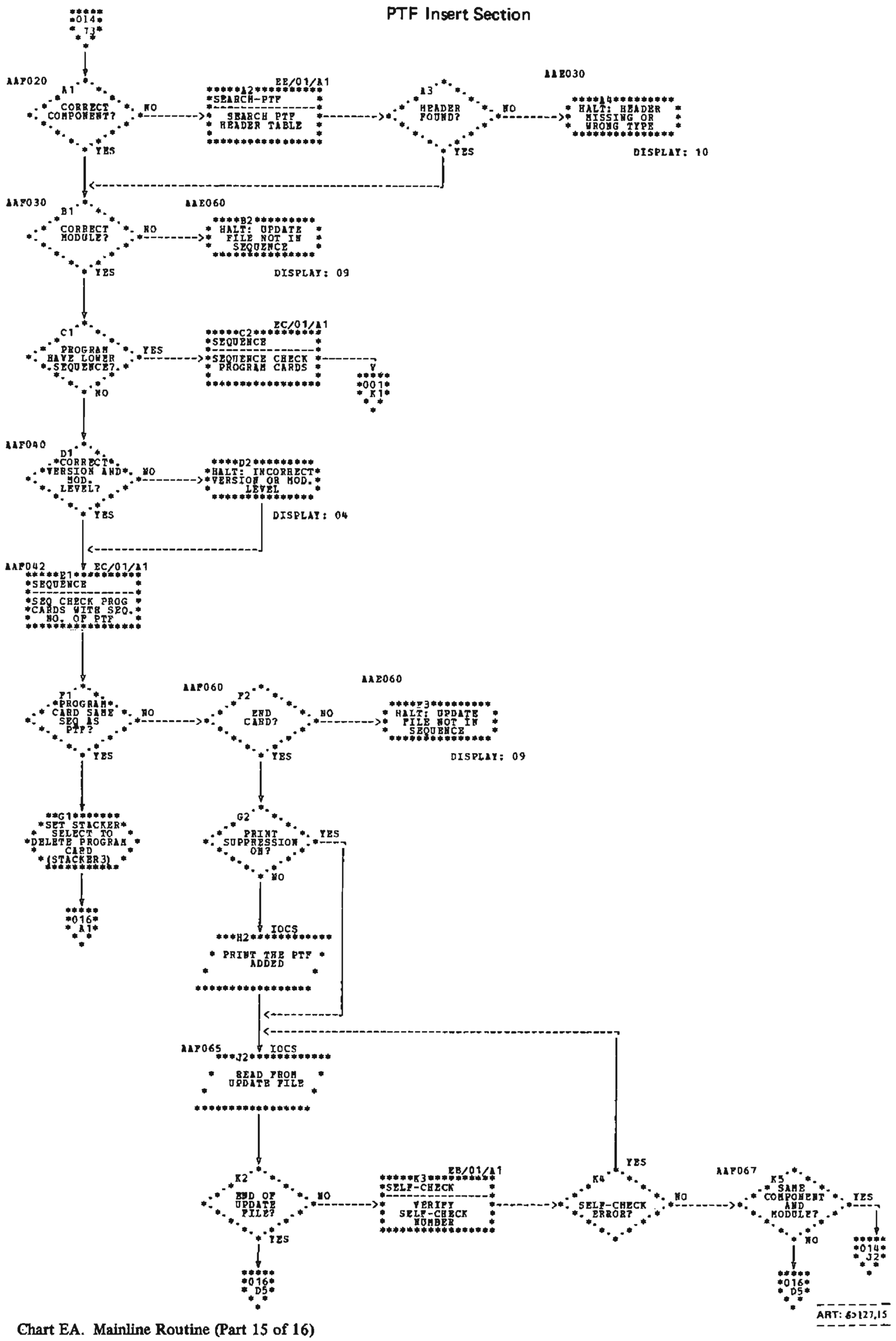
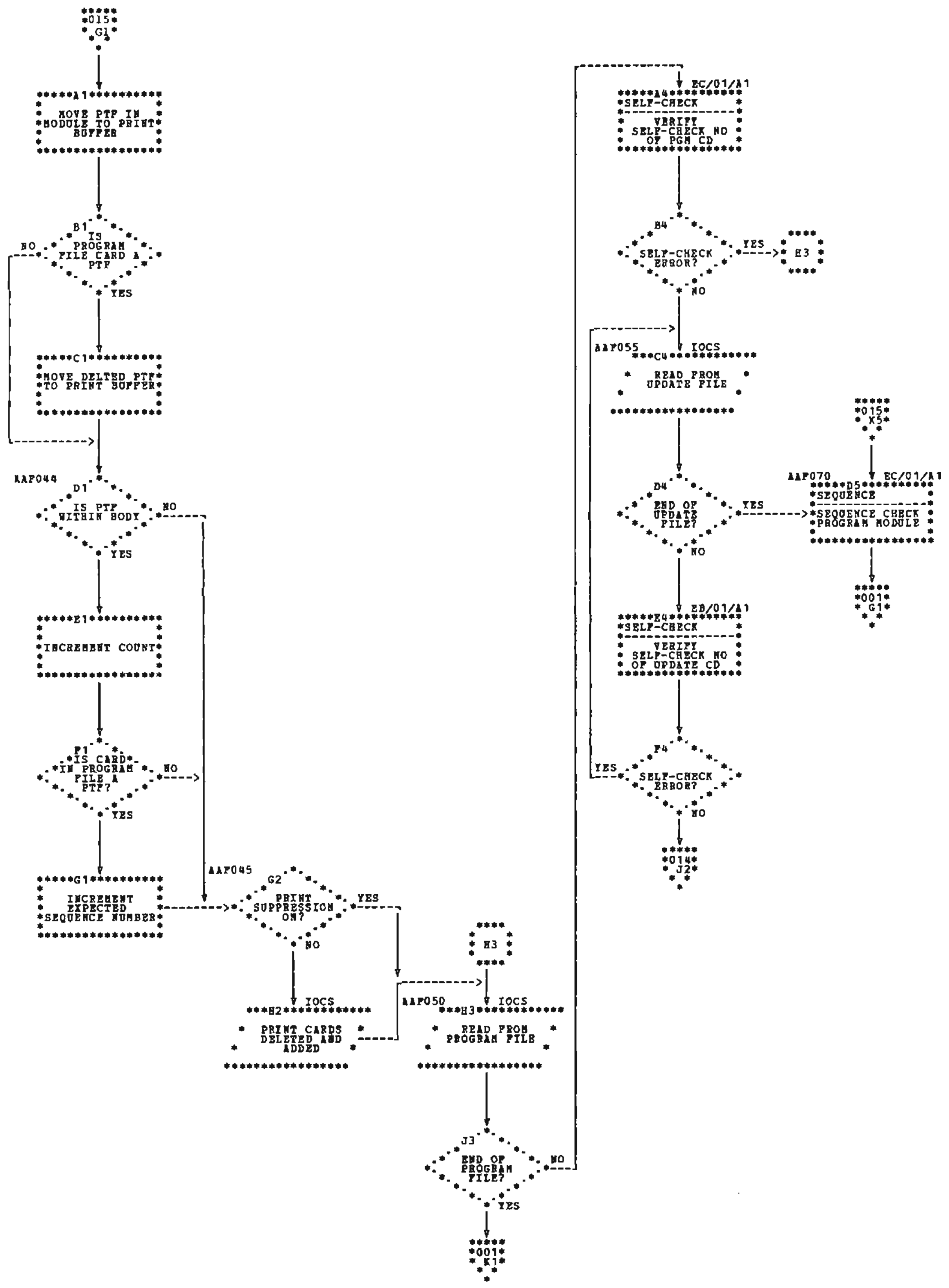

014 J3
AAF020 A1 CORRECT COMPONENT? NO YES
EE/01/A1 A2 SEARCH-PTF SEARCH PTF HEADER TABLE
A3 HEADER FOUND? NO YES
AAE030 A4 HALT: HEADER MISSING OR WRONG TYPE
DISPLAY: 10
AAF030 B1 CORRECT MODULE? NO YES
AAE060 B2 HALT: UPDATE FILE NOT IN SEQUENCE
DISPLAY: 09
C1 PROGRAM HAVE LOWER SEQUENCE? YES NO
EC/01/A1 C2 SEQUENCE SEQUENCE CHECK PROGRAM CARDS
001 K1
AAF040 D1 CORRECT VERSION AND MOD. LEVEL? NO YES
D2 HALT: INCORRECT VERSION OR MOD. LEVEL
DISPLAY: 04
AAF042 EC/01/A1 E1 SEQUENCE SEQ CHECK PROG CARDS WITH SEQ. NO. OF PTF
F1 PROGRAM CARD SAME SEQ AS PTF? NO YES
AAF060 F2 END CARD? NO YES
AAE060 F3 HALT: UPDATE FILE NOT IN SEQUENCE
DISPLAY: 09
G1 SET STACKER SELECT TO DELETE PROGRAM CARD (STACKER3)
016 A1
G2 PRINT SUPPRESSION ON? YES NO
IOCS H2 PRINT THE PTF ADDED
AAF065 IOCS J2 READ FROM UPDATE FILE
K2 END OF UPDATE FILE? NO YES
016 D5
EB/01/A1 K3 SELF-CHECK VERIFY SELF-CHECK NUMBER
K4 SELF-CHECK ERROR? NO YES
AAF067 K5 SAME COMPONENT AND MODULE? YES NO
014 J2
016 D5
ART: 65127.15

Chart EA. Mainline Routine (Part 15 of 16)



ART: 55127.16

Chart EA. Mainline Routine (Part 16 of 16)

Self-Check Routine

Entry Point: AAK010

Chart: EB

Functions:

- Moves the 3-byte self-check number from the card image area to a save area and sets the three bytes in the card image area to binary zeros.
- Converts the 3-byte self-check number using the low order six bits of each of the three bytes and ignoring the low order two bits of the result to obtain a 2-byte number.
- Calculates a 2-byte self-check number as follows:
  1. Cumulatively "exclusively ORs" each odd numbered card column with a 1-byte hold area initially set to binary zeros.
  2. Cumulatively "exclusively ORs" each even numbered card column with a 1-byte hold area initially set to binary zeros.
- Compares the self-check number on the card with the calculated self-check number (the odd numbered column result is used as the high order position and the even numbered column result as the low order position) and halts if not equal.

Subroutines Used: None

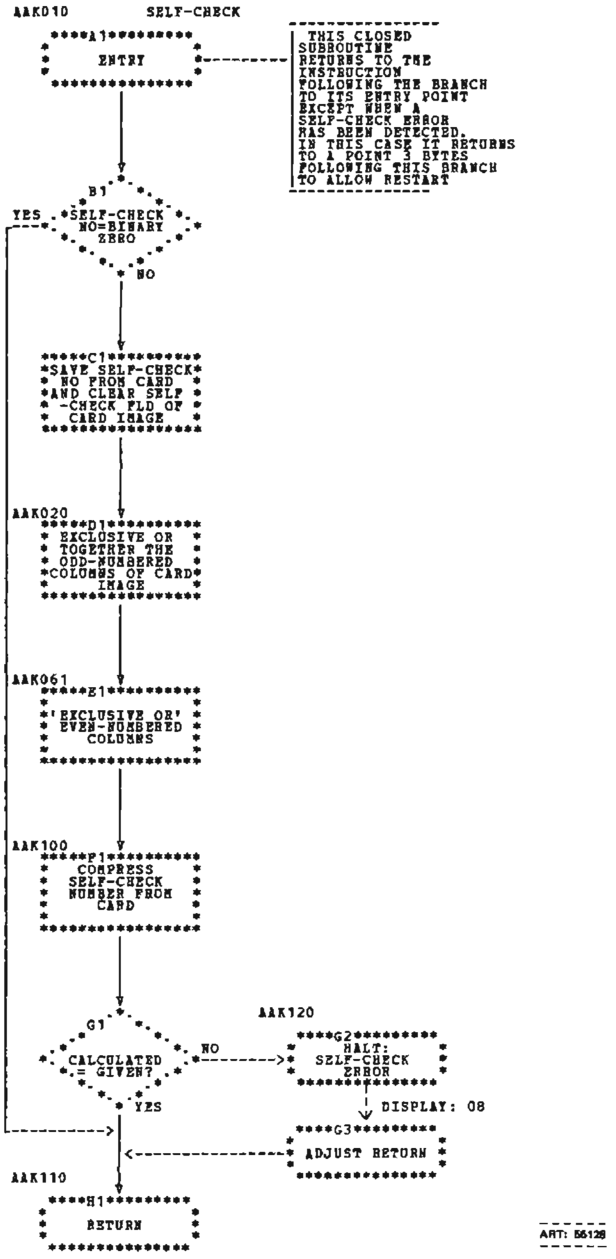

AAK010 SELF-CHECK
A1 ENTRY
THIS CLOSED SUBROUTINE RETURNS TO THE INSTRUCTION FOLLOWING THE BRANCH TO ITS ENTRY POINT EXCEPT WHEN A SELF-CHECK ERROR HAS BEEN DETECTED. IN THIS CASE IT RETURNS TO A POINT 3 BYTES FOLLOWING THIS BRANCH TO ALLOW RESTART
B1 SELF-CHECK NO=BINARY ZERO
YES
NO
C1 SAVE SELF-CHECK NO FROM CARD AND CLEAR SELF-CHECK FLD OF CARD IMAGE
AAK020
D1 EXCLUSIVE OR TOGETHER THE ODD-NUMBERED COLUMNS OF CARD IMAGE
AAK061
E1 'EXCLUSIVE OR' EVEN-NUMBERED COLUMNS
AAK100
F1 COMPRESS SELF-CHECK NUMBER FROM CARD
G1 CALCULATED = GIVEN?
NO
YES
AAK120
G2 HALT: SELF-CHECK ERROR
DISPLAY: 08
G3 ADJUST RETURN
AAK110
H1 RETURN
ART: 56128

Chart EB. Self-Check Routine

Sequence Check Routine

Entry Point: AAL010

Chart: EC

Functions:

- Prints module and component IDs and audit trail headings.
- Reads a card from the Program File or from the Update File in the case of a replacement. XR1 is passed to this routine as a parameter and points to the desired buffer when this routine is entered.
- Finds component header information in the Component Header Table via the Component Table Look-Up routine.
- Checks the sequence of an entire module by:

  1. Component ID and module ID. The data on the card being checked is compared to the data in the Component Table Hold area (COMPHD).
  2. Sequence number. Cards must be in ascending consecutive order within the body of a module deck. (The body of a deck consists of all the cards of that deck with the exception of the header card, end card, and PTF cards immediately preceding the end card.) Cards which are outside the body of a deck must be in ascending order.

- Counts the cards in a module.

At the end of every module, the Sequence Check routine performs the following functions:

- Compares the accumulated card count of the module against the module card count on the module header card.
- Lists the module card count from the header card and the accumulated card count of the module.

  1. If the counts do not agree, the listing is flagged with asterisks, and a halt occurs. Processing resumes when the operator presses START.
  2. If the counts agree, the module card count is added to the component card count and processing continues. An exception is the RPG Compiler with one or more user subroutines (EB phases). These phases are not counted in the component card count, but are sequence checked and checked for correct module card count.

- Resets expected sequence number.

Subroutines Used:

- Read Only - Both Hoppers (Object) MFCU IOCS
- Line Printer - Single Feed Carriage (Object) Printer IOCS
- Self-Check Routine

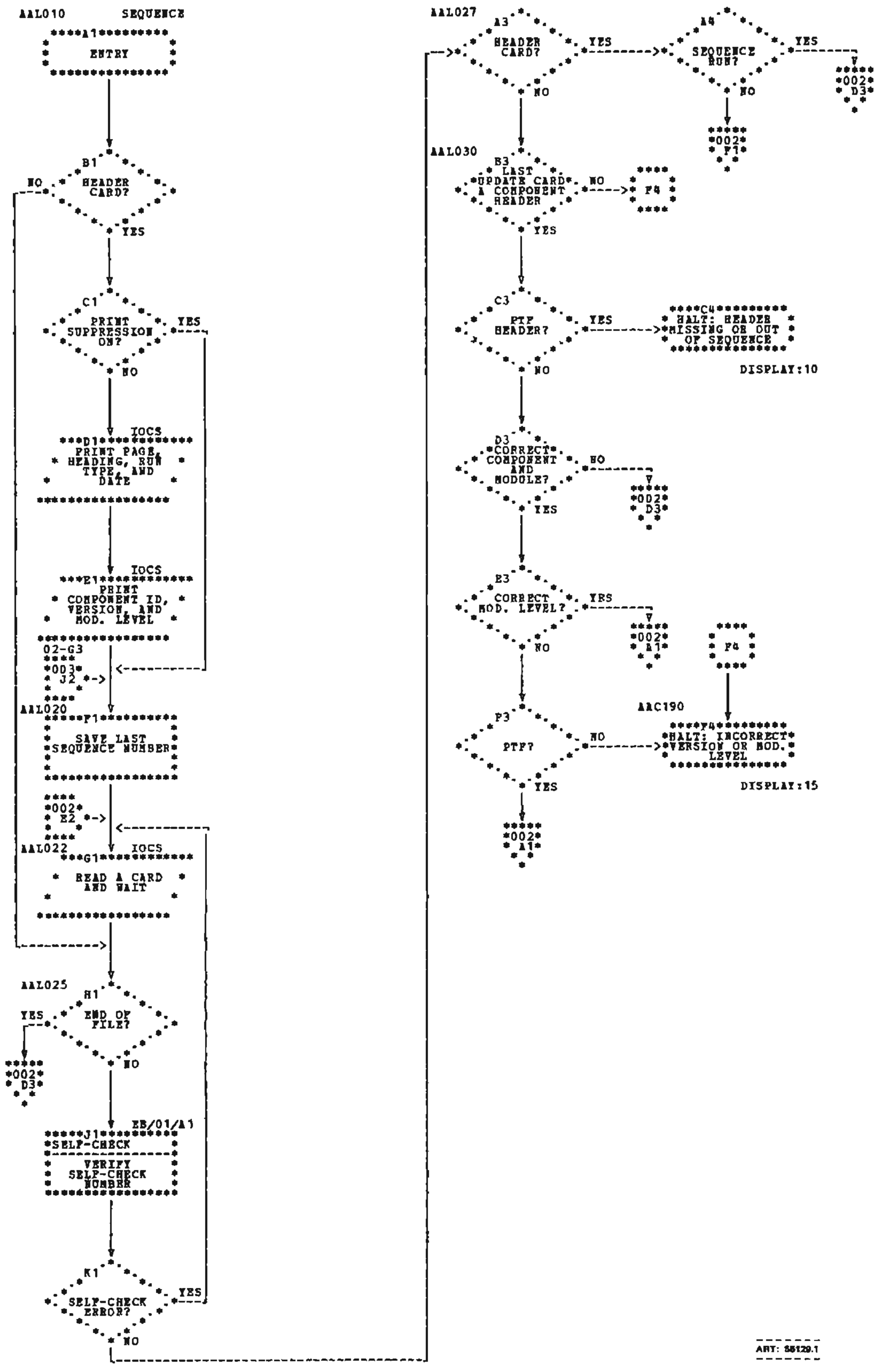


Chart EC. Sequence Check Routine (Part 1 of 3)

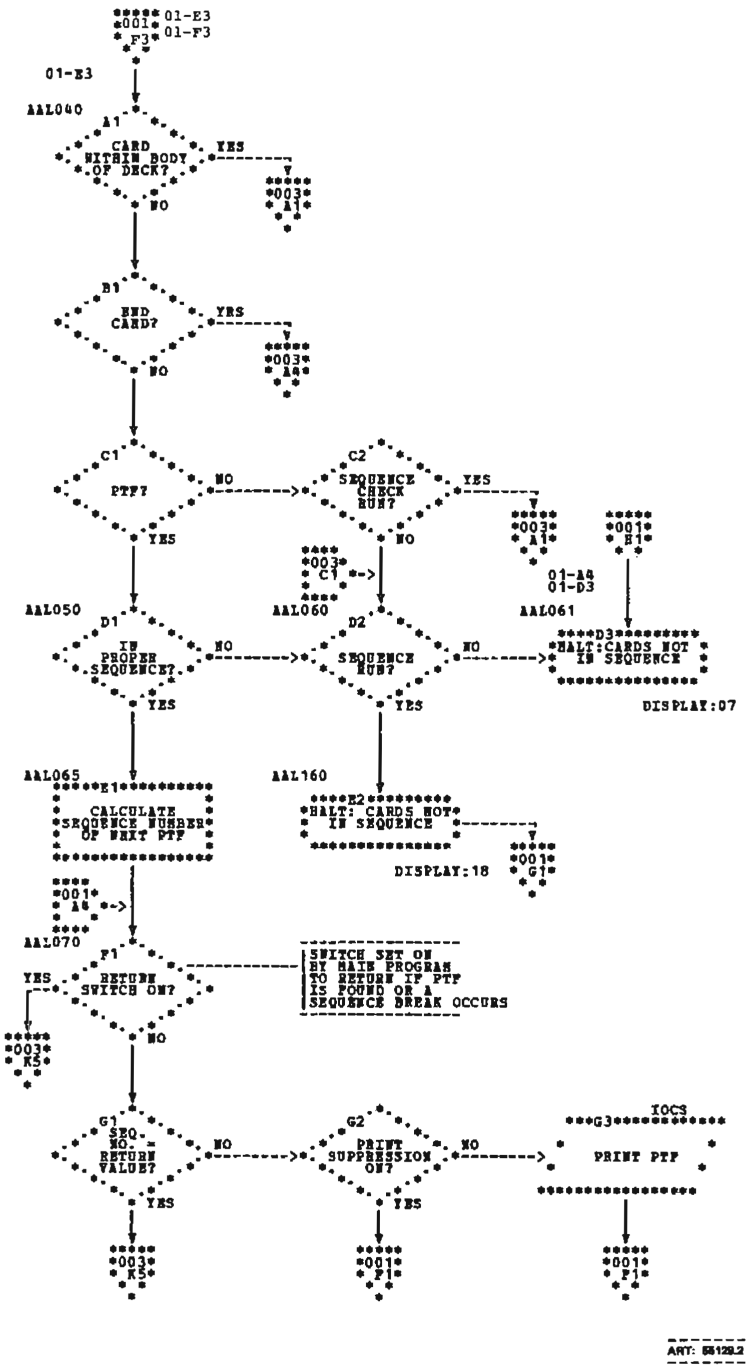


Chart EC. Sequence Check Routine (Part 2 of 3)

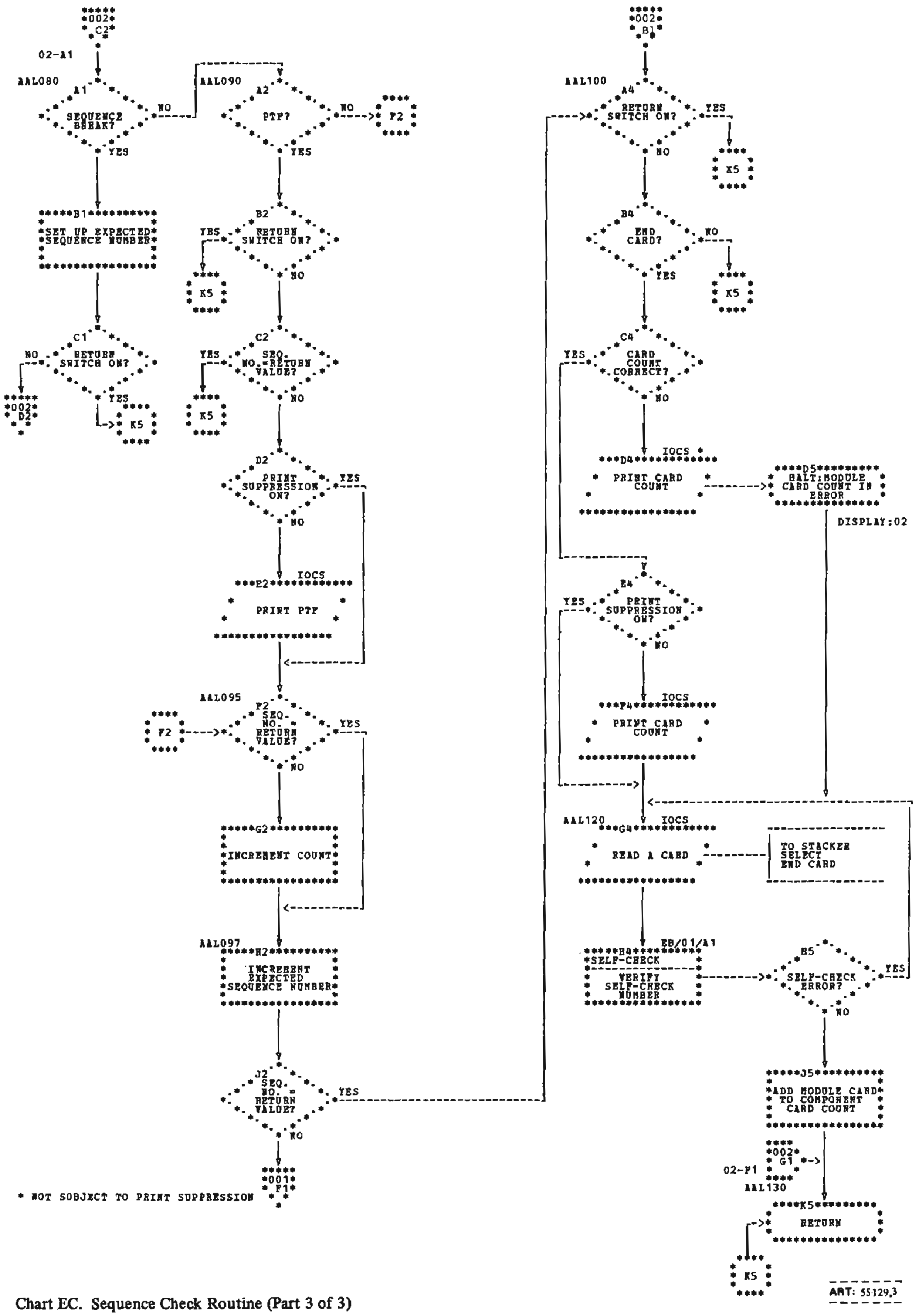


Chart EC. Sequence Check Routine (Part 3 of 3)

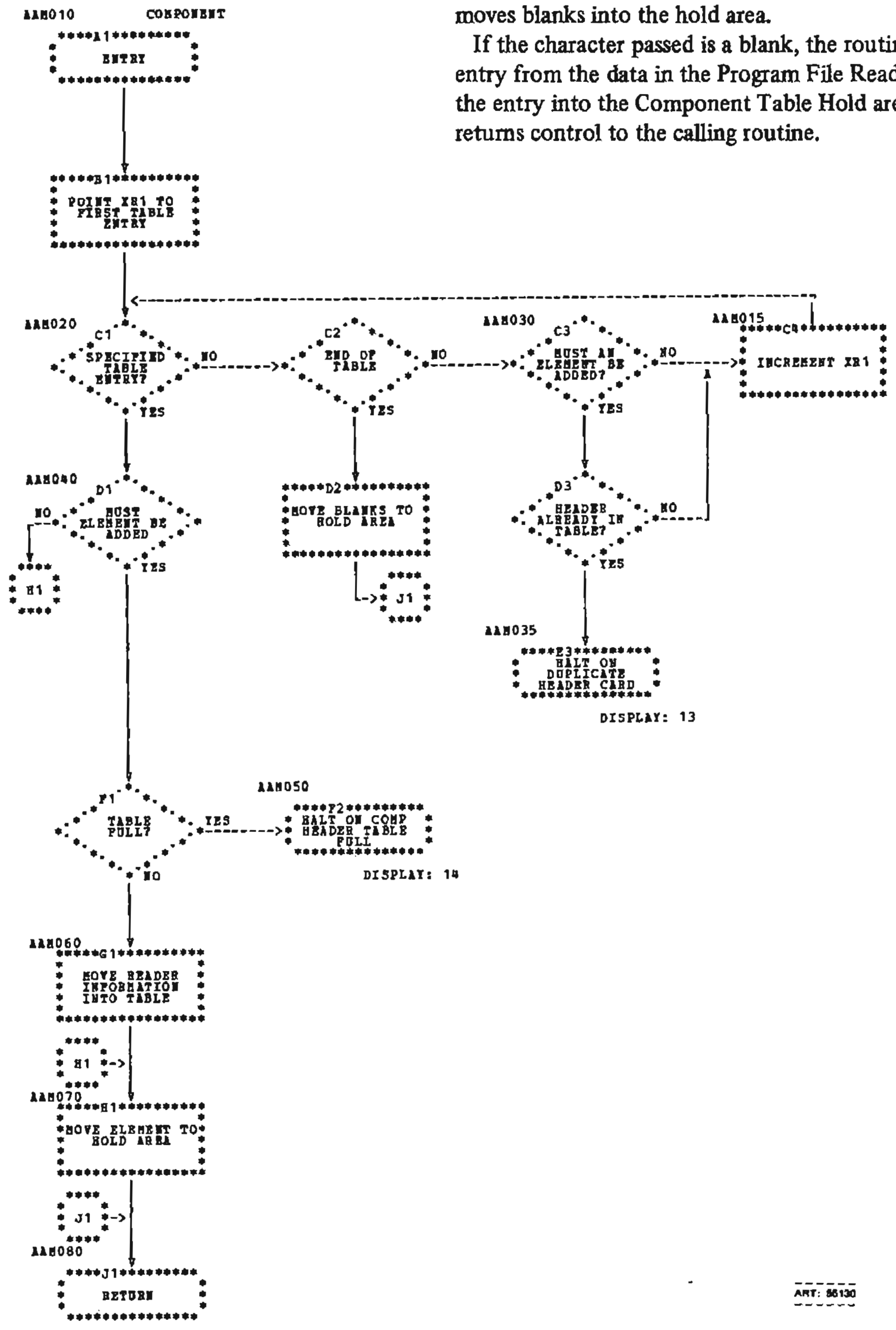
Component Table Look-Up Routine

Entry Point: AAM010

Chart: ED

Functions: This routine inserts information into or accesses information from the Component Header Table. This is accomplished by the use of parameters. If the calling routine passes a character into a Compare Logical Immediate instruction, the routine uses it as the component ID for which to search. If it is found, the routine moves the entry into the Component Table Hold area (COMPHD) and returns to the calling routine. If it is not found, the routine moves blanks into the hold area.

If the character passed is a blank, the routine builds an entry from the data in the Program File Read buffer, moves the entry into the Component Table Hold area, and then returns control to the calling routine.



ART: 85130

Chart ED. Component Table Look-Up Routine

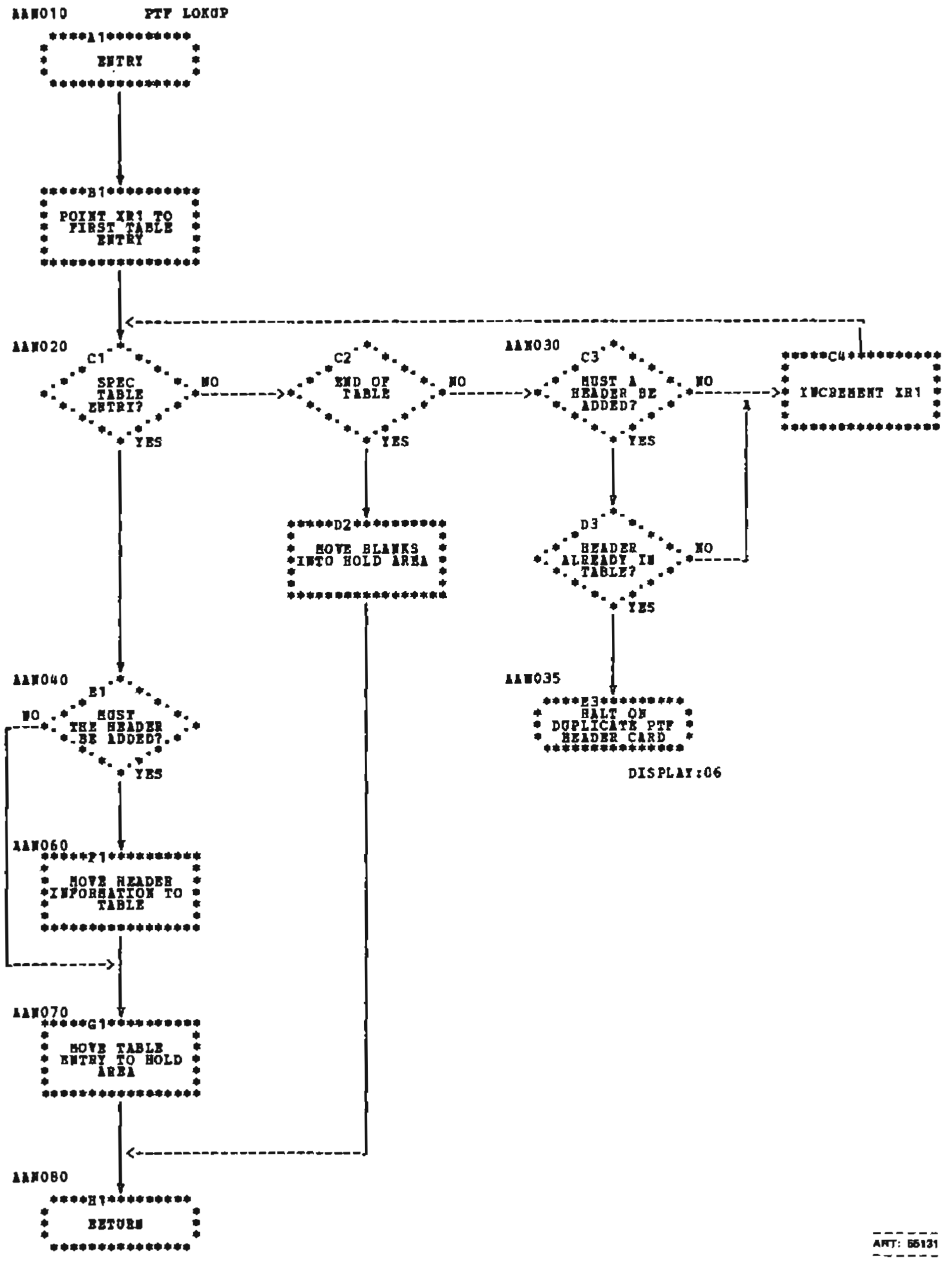
PTF Table Look-Up Routine

Entry Point: AAN010

Chart: EE

Functions: This routine inserts information into and accesses information from the PTF Header Table. This is accomplished by the use of parameters. If the calling routine passes a character into a Compare Logical Immediate instruction, the routine uses it as the component ID for which to search. If it is found, the routine moves the entry into the PTF Hold area (PTFHD) and returns to the calling routine. If it is not found, the routine moves blanks into the PTF Table Hold area.

If the character passed is a blank, it builds an entry from the information in the Update File Read Buffer, moves the entry into the PTF Table Hold area and returns control to the calling routine.



AAN010 PTF LOKUP
A1 ENTRY
B1 POINT XR1 TO FIRST TABLE ENTRY
AAN020 C1 SPEC TABLE ENTRY?
NO
YES
C2 END OF TABLE
NO
YES
AAN030 C3 MUST A HEADER BE ADDED?
NO
YES
C4 INCREMENT XR1
D2 MOVE BLANKS INTO HOLD AREA
D3 HEADER ALREADY IN TABLE?
NO
YES
AAN035 E3 HALT ON DUPLICATE PTF HEADER CARD
DISPLAY:06
AAN040 E1 MUST THE HEADER BE ADDED?
NO
YES
AAN060 F1 MOVE HEADER INFORMATION TO TABLE
AAN070 G1 MOVE TABLE ENTRY TO HOLD AREA
AAN080 H1 RETURN
ART: 55131

Chart EE. PTF Table Look-Up Routine

Section 4. Data Area Formats

Update File Read Buffer (UPDATE)

This 96-byte area, referenced by the label UPDATE, is a read buffer and work area for the Update File. Data information in this area may vary depending upon the type of run. (For card formats see Figure 2-1).

Program File Read Buffer (PRGRD)

This 96-byte area, referenced by the label PRGRD, is the read buffer and work area for the Program File (For card formats see Figure 2-1).

Component Header Table (COMPTB)

This is a 260-byte data area referenced by the label COMPTB. It is a table of component header information that is comprised of twenty 13-byte members which are initialized to blanks. The end of the table is indicated by a b̸/. Each member is comprised of the same data found in the Component Table Hold area, shown in Figure 2-5.

Component Table Hold Area (COMPHD)

This 13-byte data area, referenced by the label COMPHD, contains information inserted into or from the Component Header Table. The format of this hold area is shown in Figure 2-5.

PTF Header Table (PTFTAB)

This 100-byte data area is referenced by the label PTFTAB. It is comprised of twenty 5-byte members. The table is initialized to blanks and the end of the table is denoted by a b̸/. It is a table of PTF header card information. Each member is comprised of the same data found in the PTF Table Hold area.

PTF Identification Table (IDTAB)

This 65-byte area, referenced by the label IDTAB, is initialized to binary zeros. The first 64 bytes contain the PTF numbers (cc 90) from the PTF cards of the module. The last byte contains binary zeros and is used for clearing the first 64 bytes.

PTF Table Hold Area (PTFHD)

This 5-byte data area, referenced by the label PTFHD, contains information to be inserted into or received from the PTF Header Table. The format of this hold area is as follows:

Label	*Location*	*Length*	*Definition*
PHDID	PTFHD-4	1	Component ID
PHDVER	PTFHD-2	2	Component Version
PHDLVL	PTFHD	2	Component Modification Level

Print Buffer (PRTBUF)

This is a 128-byte data area which is referenced by the label PRTBUF. It is the print buffer for the printer.

COMPHD-12
COMPHD

Label	Location	Length	Definition
CHDID	COMPHD-12	1	Component ID
CHDCNT	COMPHD-8	4	Component Card Count
CHDVER	COMPHD-6	2	Component Version
CHDLVL	COMPHD-4	2	Component Modification Level
CHDCAL	COMPHD	4	Component Calculated Card Count

Figure 2-5. Component Table Hold Area (COMPHD) Format

Name (NAME)

This 11-byte storage area saves the name of the program deck being maintained. The data in this area is in a character format and is initialized to blanks.

Module ID (MODID)

This 2-byte storage area saves the Program File module ID for module headers read from the Program File. The data in this area is in a character format and is initialized to blanks.

Module Currency Level (MODCUR)

This is a 2-byte storage area for the module modification level contained on the module header card. The data in this area is in zoned decimal format and is initialized to blanks.

Module Card Count (MODCNT)

This 3-byte data area stores the module card count that is located on the module header card. The data in this area is in zoned decimal format and is initialized to zeros. This count is printed out as "EXPECTED CARD COUNT."

Module Calculated Card Count (MODCAL)

This is a 3-byte storage area used to calculate the module card count of a given module. The data in this area is in zoned decimal format and is initialized to zeros.

This count is printed out as "ACTUAL CARD COUNT." The count includes all cards from the header card through the end card, except PTFs that are additions to the deck. PTF cards that have replaced cards within the module are included in the count.

Expected Sequence Number (EXPSEQ)

This 3-byte storage area holds the expected sequence number of the program card being checked during the sequence check. The data in this area is in zoned decimal format and is initialized to *001*.

Body Count (BODYCT)

This is a 3-byte hold area for the sequence number of the last card in the body of a module. The data in this area is in zoned decimal format and is initialized to zeros.

Section 1. Introduction

The primary purpose of the IBM System/3 Device Counter Logout program is to report information about errors that were recorded during the execution of a Binary Synchronous Communications (BSC) program. This program should be run immediately after every BSC program. If it is run at any other time, invalid address processor checks may occur or invalid information may be reported. The program performs the following functions:

- Prints a listing, if requested, with an identifying heading, the date, and a list of each of the 13 BSC counters and their values.
- Punches and prints a card containing an identifying heading, the date, and the decimal value of each of the 13 BSC counters. The program inserts this card before the end card of the program.

Punched Card

A card is punched by the Device Counter Logout program and inserted prior to the program end card. The contents of this card are:

Column	*Contents*
1	C
2	Blank
3-6	Text blocks sent (in hexadecimal)
7	Blank
8-11	Text blocks received (in hexadecimal)
12	Blank
13-16	NAKS received (in hexadecimal)
17	Blank
18-21	Data checks (in hexadecimal)
22	Blank
23-26	Forward aborts (in hexadecimal)
27	Blank
28-31	Aborts (in hexadecimal)
32	Blank
33-36	Adapter checks on transmit (in hexadecimal)
37	Blank
38-41	Adapter checks on receive (in hexadecimal)
42	Blank
43-46	Invalid replies (in hexadecimal)
47	Blank
48-51	ENQs received (in hexadecimal)
52	Blank
53-56	Lost data count (in hexadecimal)
57	Blank
58-61	Disconnect timeouts (in hexadecimal)
62	Blank
63-66	Timeouts during receive data (in hexadecimal)
67	Blank
68-73	Date (mmddyy)
74-96	Blank

See Section 3, *Data Area Formats*, for an explanation of each of the counters.

System Requirements

The Device Counter Logout program requires an:

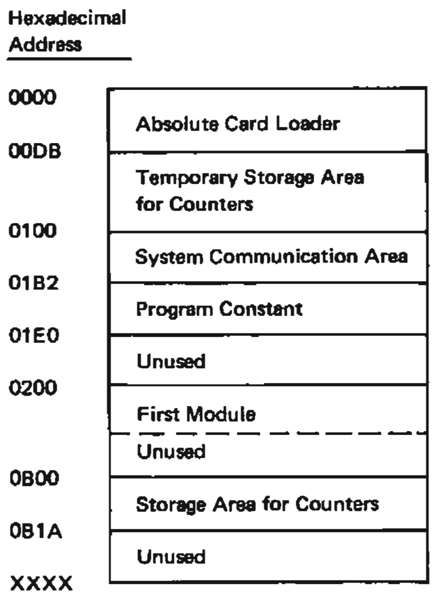
- IBM 5410 Processing Unit with a minimum of 8,196 bytes of storage
- IBM 5424 Multi-Function Card Unit (MFCU)
- IBM 5203 Printer
- IBM Binary Synchronous Communications Adapter (BSCA)

Section 2. Program Organization

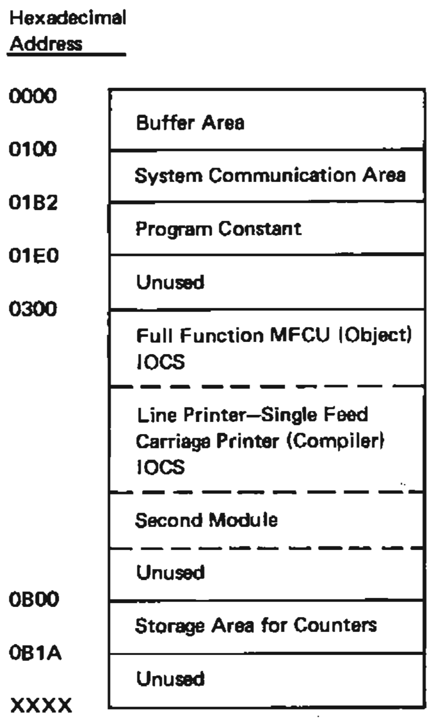
The Device Counter Logout program is made up of two modules. The first module locates the 13 counters in storage. The counters are then moved to a high storage location so they will not be overlaid by the second module. Figure 3-1 shows a storage map of the first module.

The second module converts the counters to decimal and, if requested, prints a listing of each type of counter and its value. A card is then punched containing the counter information. This punched card is merged into the program deck prior to the end card. Figure 3-2 shows a storage map of the second module.

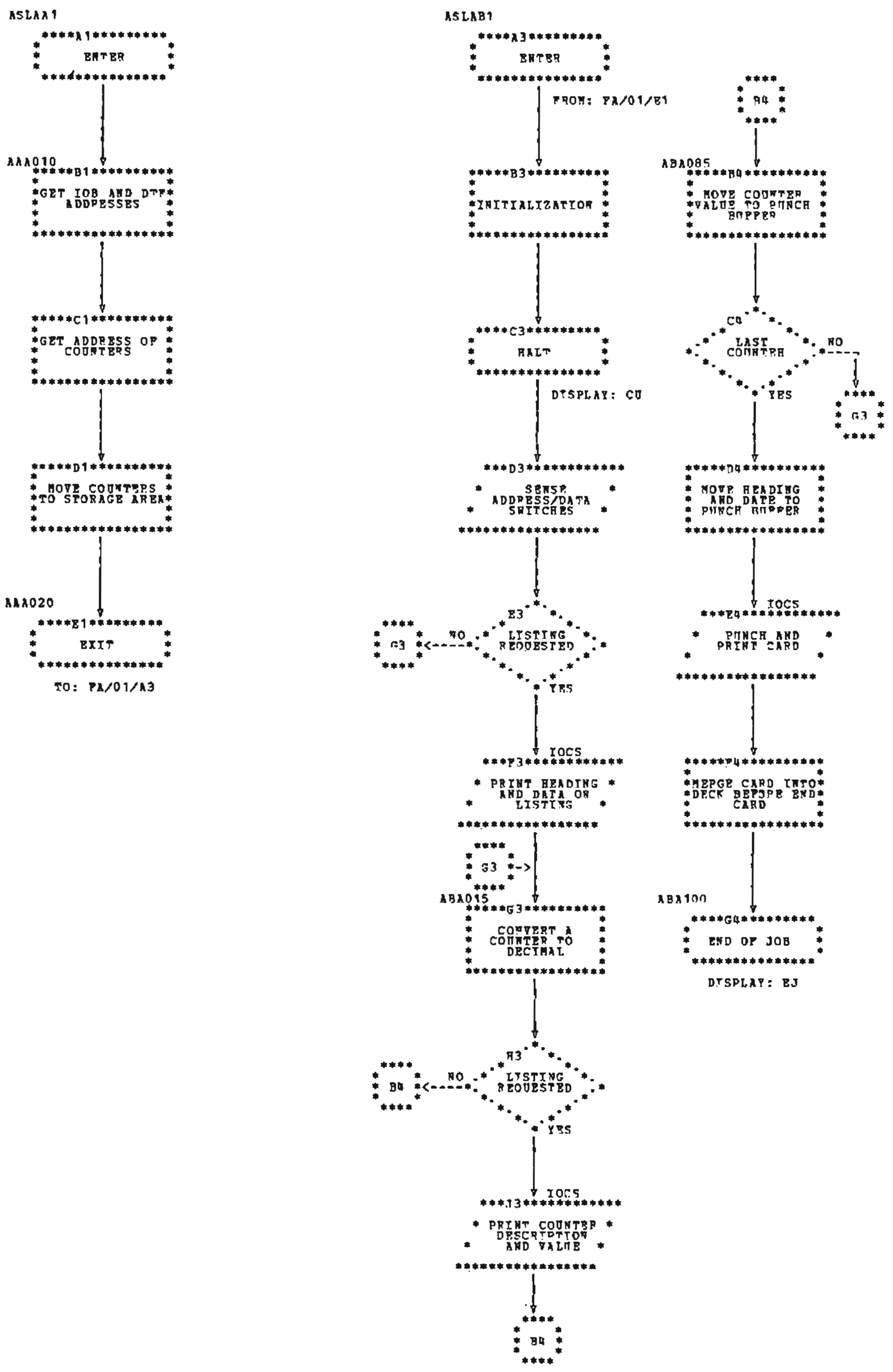
Chart FA shows a flowchart of the program.



● Figure 3-1. Storage Map for First Module of Device Counter Logout Program


Hexadecimal Address
0000
Buffer Area
0100
System Communication Area
01B2
Program Constant
01E0
Unused
0300
Full Function MFCU (Object) IOCS
Line Printer—Single Feed Carriage Printer (Compiler) IOCS
Second Module
Unused
0B00
Storage Area for Counters
0B1A
Unused
XXXX

● Figure 3-2. Storage Map for Second Module of Device Counter Logout Program



● Chart FA. Device Counter Logout

Storage Area for Counters

The 26 bytes of counters are moved to X'B00' by the first module of the Device Counter Logout program. Each of the 13 counters is two bytes long. The counters and their meanings are:

- Text blocks sent—Number of blocks of data transmitted successfully from this terminal to a remote terminal.
- Text blocks received—Number of blocks of data received successfully by this terminal from a remote terminal.
- NAKS received—Number of negative responses received by this terminal in response to data transmitted by this terminal.
- Data checks—Number of text blocks received with invalid error check bits.
- Forward aborts—Number of times a remote transmitting terminal has terminated transmission abnormally.
- Aborts—Number of times a remote receiving terminal has terminated transmission abnormally.
- Adapter checks on transmit—Number of times the following errors occurred while the terminal was transmitting data:

  1. Parity check within the adapter.
  2. Cycle steal overrun.
  3. Local storage register or control register check.

- Adapter checks on receive—Number of times the following errors occurred while the terminal was receiving data:

  1. Parity check within the adapter.
  2. Cycle steal overrun.
  3. Local storage register or control register check.

- Invalid replies—Number of abnormal responses (including no responses) from the remote terminal.
- ENQs received—Number of requests for retransmission of this terminal's last acknowledgement after the acknowledgement has already been sent.
- Lost data count—Number of text blocks received which do not fit into the receive area.
- Disconnect timeouts—Number of times the data set has dropped ready status after that status was set on.
- Timeouts during receive data—Number of times this terminal expected to receive text but did not receive anything for 3.25 seconds.

The directory lists the programs discussed in this publication for reference to the program listings on microfiche.

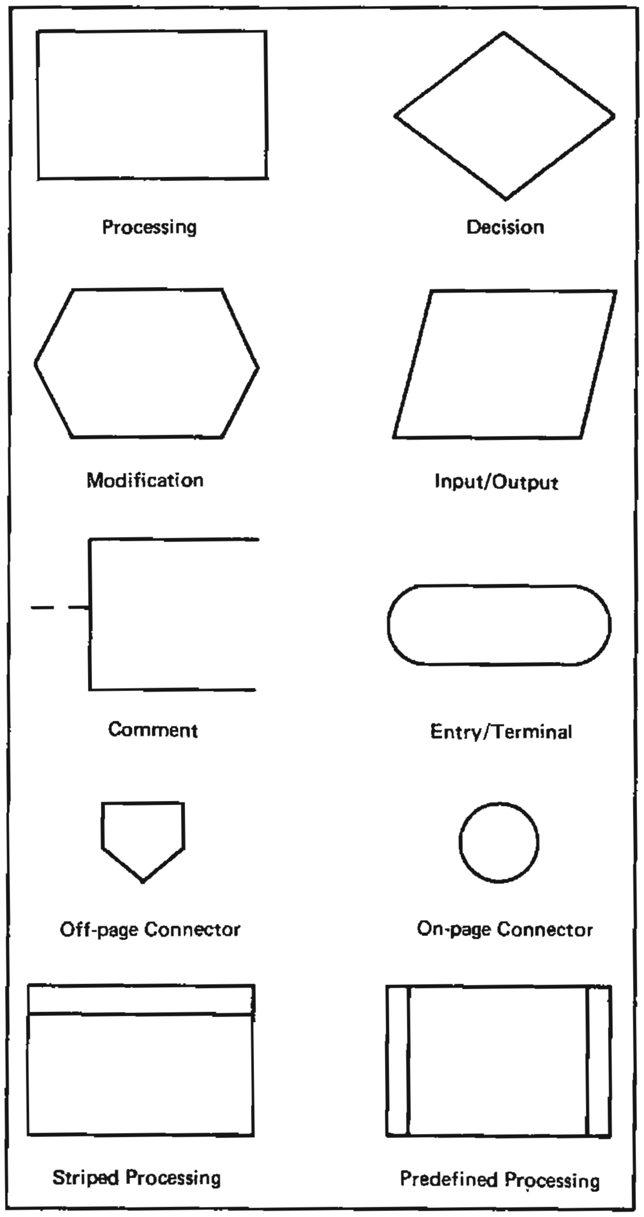
Descriptive Name	*Entry Point*	*Synopsis*
System Initialization	ASIAA1	Initializes the System Communication Region.
Program Maintenance	APMAA1	Updates program decks to maintain accuracy of system programs.
Device Counter Logout	ASLAA1	Reports information about errors that were recorded during the execution of a BSC program.

Appendix B. Flowcharting Techniques

Flowcharts are identified in the following manner:

- A flowchart that consists of only one page is identified with a chart ID of, for example, AA.
- A flowchart that consists of multiple pages with a chart ID of AA is identified as: first page = AA-01, second page = AA-02, and so on.
- A sequence of related flowcharts are identified as: first flowchart = AA, second flowchart = AB, third flowchart = AC, and so on.
- A sequence of flowcharts, each flowchart having multiple pages, are identified as: first flowchart with multiple pages = AA-01, AA-02, and so on; second flowchart with multiple pages = AB-01, AB-02, and so on; continue through the sequence of flowcharts.

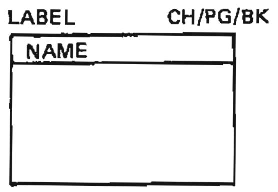
The flowcharting symbols used are:



Most of the symbols are self-explanatory but some need more explanation.

1. The striped processing block indicates the entry of a routine flowcharted in this logic manual.

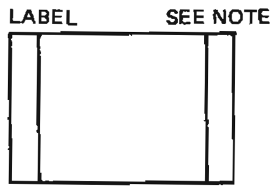
   Example:



   CH/PG/BK indicates the chart ID, page, and block identification of the flowcharted routine.

2. The predefined processing block indicates a routine flowcharted and/or described in another logic manual.

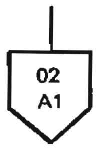
   Example:



   Note: See "logic manual title and order number"

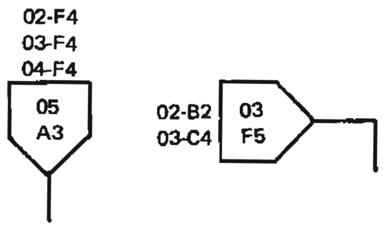
3. Off-page connectors are used to reference between different pages of the same chart ID. Off-page connectors leaving a page contain the page number and block number of their destination.

   Example:



   Off-page connectors entering a page contain the page number and block number of their origin. If the entry point referenced by the off-page connector is referenced from more than one origin, all origins are given. The origins are listed in alphameric order with the last reference contained within the block.

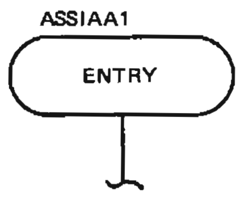
   Example:



   On-page connectors contain the location of a block on the same page. On-page connectors always contain the location of the destination block.

4. 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:



Index

System Initialization

Program Maintenance

Device Counter Logout

SY21-0522-2

IBM System/3 Printed in U.S.A. SY21-0522-2

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

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