IBM System/3
Card System
Absolute Card Loader and Input/Output Control System
Logic Manual

Second Edition (June 1971)

This is a major revision of, and obsoletes, SY21-0521-0 and Technical Newsletter SN21-7522. Flowcharts have been redrawn to current flowcharting standards, layout has been revised in some places to enhance usability, and miscellaneous changes have been made throughout the book. Changes are indicated by a vertical line at the left of the change.

This edition applies to version 01, modification 05, of IBM System/3 Card System, Program Number 5701-SC1, and to all subsequent versions and modifications until 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.

This Program Logic Manual provides documentation for maintenance of the IBM System/3 Card System Absolute Card Loader and Input/Output Control System (IOCS). This publication is divided into two parts. The first part discusses the Absolute Card Loader; the second part discusses IOCS. IOCS is divided into three chapters – MFCU IOCS, Printer IOCS, and 1442 IOCS.

The following sections are included, as required, in each part depending on the size and complexity of the routine being described:

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

2. *Method of Operation* describes the data flow and functional flow of the routine 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 areas) used by each routine.

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

This publication is intended for recall and debugging. When debugging a program, 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 these 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, IBM 80-96 Conversion Program and RPG II Support for the IBM 1442 Card Read-Punch*, SC21-7518.
- *IBM System/3 Card System System Control Programs Logic Manual*, SY21-0522.

PART I
ABSOLUTE CARD LOADER

Section 1. Introduction

The IBM System/3 Absolute Card Loader program performs the following functions:

- Reads object program cards.
- Compresses text and end cards.
- Ignores any card that is neither a text nor an end card.
- Moves data from the read buffer to a specific location in storage.
- Executes the instructions on the end card and branches to the program entry point.

CONTROL INFORMATION

The Absolute Card Loader recognizes two types of cards found in the object program, the text card and the end card.

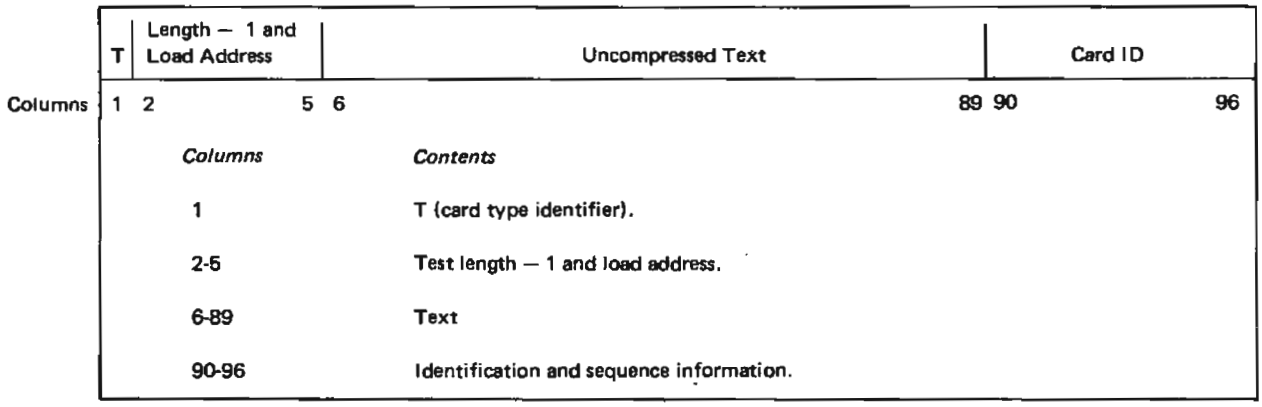
Text Card

The text cards contain instructions and constants, along with the length and address of the storage area into which the instructions and constants are to be loaded. The 96-column text card is shown in Figure 1-1.

Columns 2-89 are compressed into 65 bytes of text by a 4-column-to-3-byte compression. Figure 1-2 shows how the text card resides in storage after this compression.

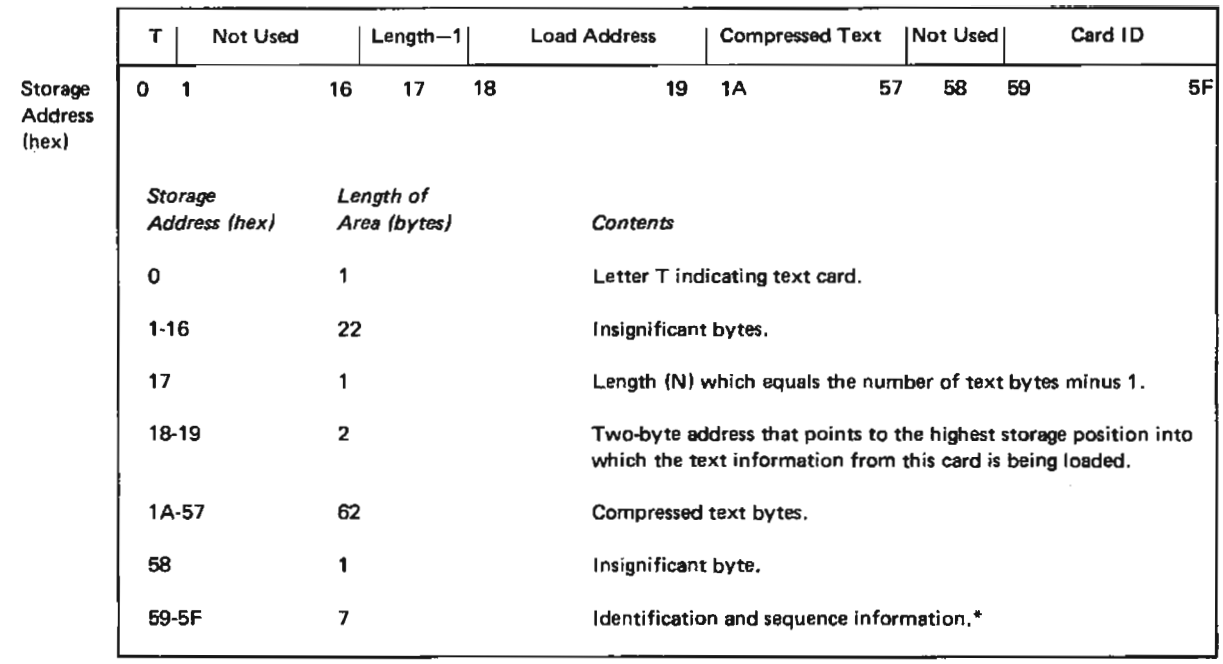
End Card

The end card contains instructions that move the local storage register (LSR) display routine to storage location X'60'-X'77'. The end card also contains instructions to clear itself from storage and transfer control to the object program just loaded. Figure 1-3 shows the format of the end card. Like the text card, columns 2-89 of the end card are compressed into 65 bytes. Figure 1-4 shows how the end card resides in storage after this compression.



Columns	*Contents*
1	T (card type identifier).
2-5	Test length — 1 and load address.
6-89	Text
90-96	Identification and sequence information.

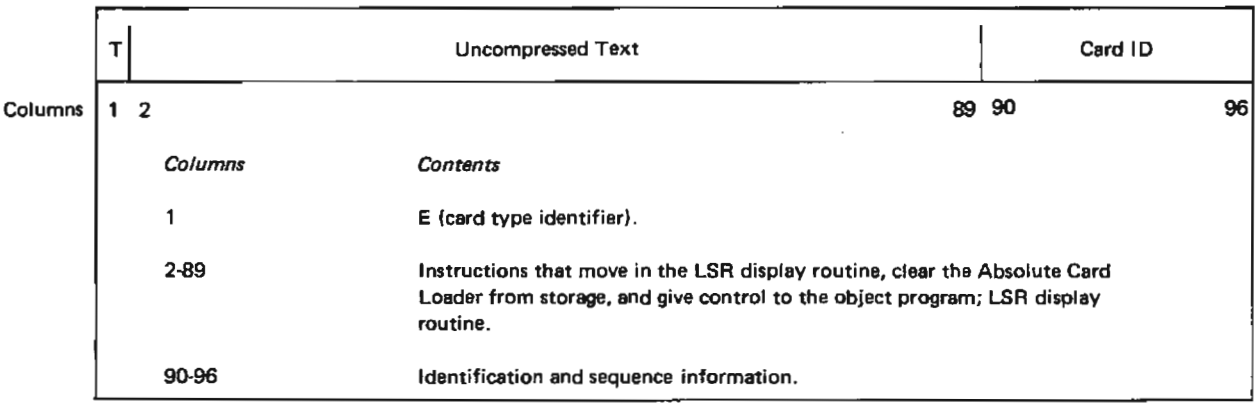
Figure 1-1. Text Card Format Before Compression



Storage Address (hex)	*Length of Area (bytes)*	*Contents*
0	1	Letter T indicating text card.
1-16	22	Insignificant bytes.
17	1	Length (N) which equals the number of text bytes minus 1.
18-19	2	Two-byte address that points to the highest storage position into which the text information from this card is being loaded.
1A-57	62	Compressed text bytes.
58	1	Insignificant byte.
59-5F	7	Identification and sequence information.*

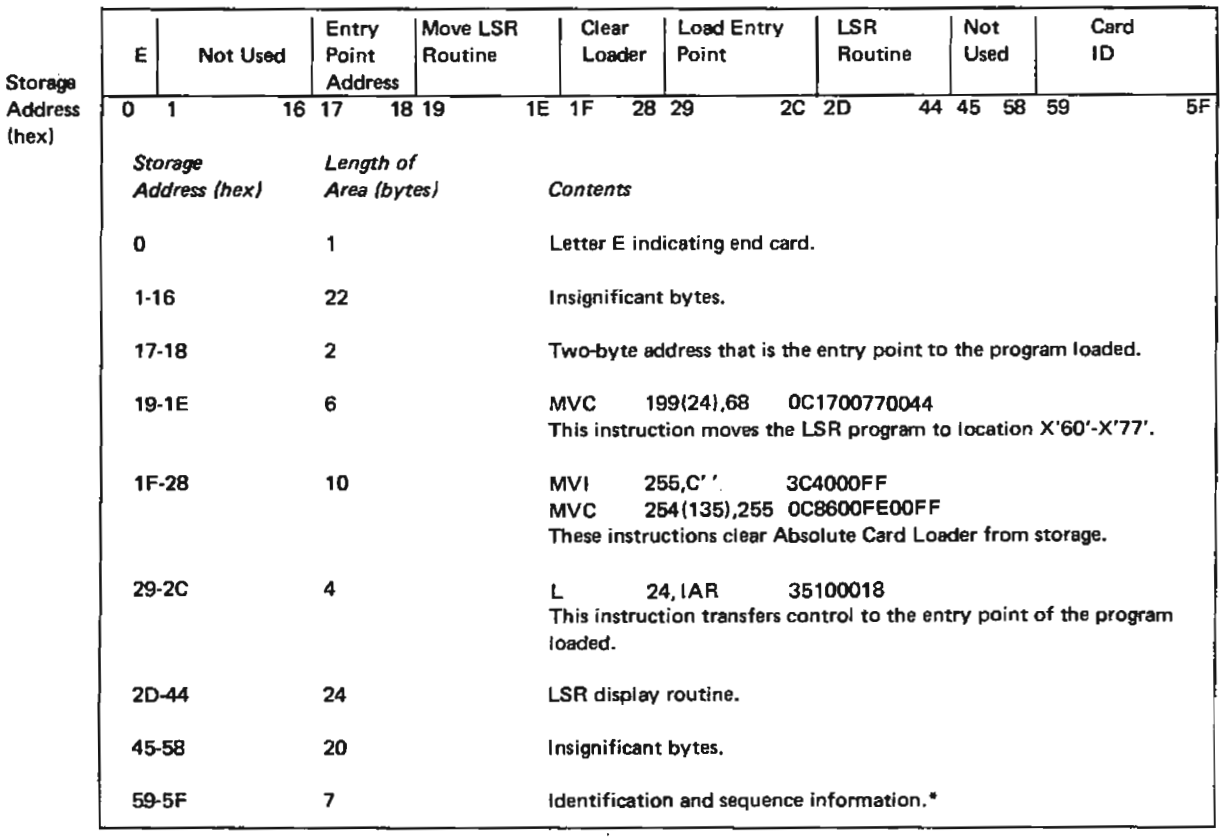
*For error recovery purposes, 3 is subtracted from the sequence number.

Figure 1-2. Image of Text Card in Storage After Compression



Columns	*Contents*
1	E (card type identifier).
2-89	Instructions that move in the LSR display routine, clear the Absolute Card Loader from storage, and give control to the object program; LSR display routine.
90-96	Identification and sequence information.

Figure 1-3. End Card Format Before Compression



Storage Address (hex)	*Length of Area (bytes)*	*Contents*
0	1	Letter E indicating end card.
1-16	22	Insignificant bytes.
17-18	2	Two-byte address that is the entry point to the program loaded.
19-1E	6	MVC 199(24),68 0C1700770044 This instruction moves the LSR program to location X'60'-X'77'.
1F-28	10	MVI 255,C' ' 3C4000FF MVC 254(135),255 0C8600FE00FF These instructions clear Absolute Card Loader from storage.
29-2C	4	L 24,IAR 35100018 This instruction transfers control to the entry point of the program loaded.
2D-44	24	LSR display routine.
45-58	20	Insignificant bytes.
59-5F	7	Identification and sequence information.*

*For error recovery purposes, 3 is subtracted from the sequence number.

Figure 1-4. Image of End Card in Storage After Compression

COMPRESSION

The data representing object program instructions is expanded before it is punched on cards; therefore, it is not in executable format. In order to reformat object program cards and produce an executable module in storage, the Absolute Card Loader compresses columns 2-89 of all text and end cards. Each 4-column segment of expanded object code on these cards is compressed by isolating the low-order six bits in each byte and forming a 3-byte segment. The 3-byte segments are then shifted to form a string of text in storage location X'17'-X'58' (Figure 1-5). Figure 1-6 is a flowchart of the compression and string-building portions of the Absolute Card Loader program. The alphameric labels on the chart blocks refer to segments of code in the program listing. These code segments perform the following operations:

AAA150-AAA160: Since two different punch combinations are read as *XX*010000, *11*010000 (X'D0') is changed to *00*101010 (X'2A'), which has no equivalent punch combination. This ensures that the low-order six bits are unique.

AAA180: Two ALC instructions are used to shift the data two bit positions to the left. Since only the low-order six bits of data represent text, the two high-order bits are removed by shifting left. XR2 (index register 2) +1 points to the rightmost byte of the 4-byte field to be compressed. The compression cycle is repeated until four bytes have been compressed into three bytes.

AAA190: The compressed text is moved to the right. The rightmost byte of the contiguous string is at storage location X'58'. When the entire card has been compressed, the string of text resides in storage location X'17'-X'58'.

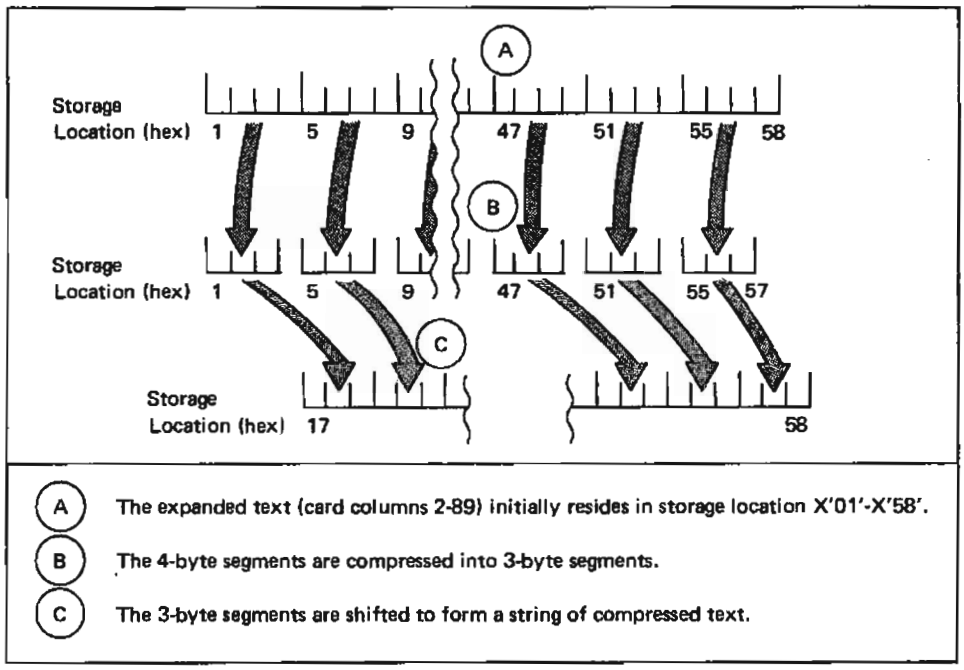


Figure 1-5. Compression of Expanded Object Code

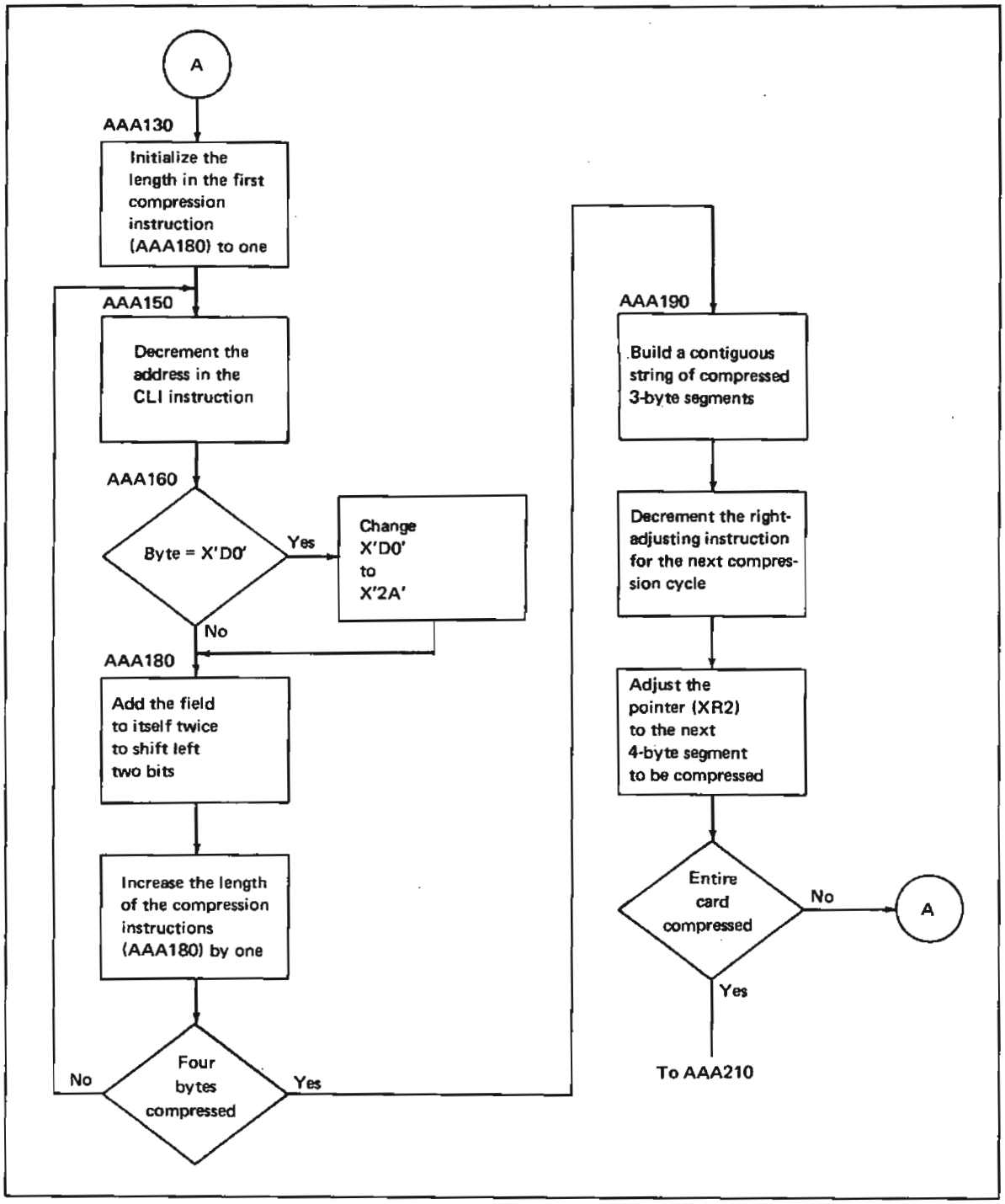
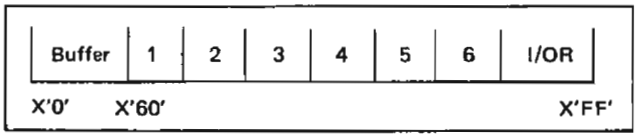


Figure 1-6. Compression of Text and End Cards

ABSOLUTE CARD LOADER BOOTSTRAP

The Absolute Card Loader consists of six cards. The first card is loaded by pressing the PROGRAM LOAD key. The instructions on this card are executed and a branch is made to the I/O routine (I/OR) to read the next card (bootstrap the next card). All six cards are read in the IPL mode. The following shows the area in storage that is built by the Absolute Card Loader:



The six cards contain the following:

Card 1: Tier 1

- Instruction to initialize XR1 (index register 1).
- Instruction to move I/O routine to I/OR.
- Instruction to move the first eight bytes of Absolute Card Loader to #1.
- Instruction to initialize the System Communication Area to at least 4K.
- Instruction to branch to I/OR to read the next card.

Tier 2

- I/O routine.
- Eight bytes to be loaded into #1.

Card 2: Tier 1

- Instruction to move Absolute Card Loader to #2.
- Instruction to branch to I/OR to read the next card.
- Part of #2.

Tier 2

- Remainder of #2.

Card 3: Tier 1

- Instructions to move Absolute Card Loader to #3.
- Instruction to branch to I/OR to read the next card.
- Part of #3.

Tier 2

- Remainder of #3.

Card 4: Tier 1

- Instructions to move Absolute Card Loader to #4.
- Instruction to branch to I/OR to read the next card.
- Part of #4.

Tier 2

- Remainder of #4.

Card 5: Tier 1

- Instructions to move Absolute Card Loader to #5.
- Instruction to branch to I/OR to read the next card.
- Part of #5.

Tier 2

- Remainder of #5.

Card 6: Tier 1

- Instructions to move the last part of the Absolute Card Loader to #6, overlaying part of I/OR.
- Instruction to move the storage size to CLI of loader from the System Communication Area.
- Instruction to branch to the entry point of the Absolute Card Loader.

Tier 2

- Remainder of #6.

LOCAL STORAGE REGISTER (LSR) DISPLAY ROUTINE

The local storage register (LSR) display routine is used to show the contents of a given register without destroying the contents of that register. This routine is moved from the object program end card to storage location X'60'-X'77' by instructions on the end card. The LSR display routine is shown in Figure 1-7.

The operator performs the following procedure to use the LSR display routine:

1. Sets the IAR to X'66'.
2. Sets the console address/data switches:
   a. The two left switches are set to 34 to store a register or to 30 to sense I/O device information.
   b. The two right switches are set to the data source to be displayed.
3. Presses the START key. A halt condition is displayed on the register display unit. The Q byte of the first halt is the high-order byte of the information being displayed.

When the START key is pressed again, another halt condition occurs. The Q byte of the second halt is the low-order byte of the information being displayed.

Another register or sense information can be displayed by changing the setting of the two rightmost console address/data switches and pressing the START key.

Location	Instructions (hex)	Instructions (symbolic)		
X'0060'	F00000	HALT1	HPL	X'00',X'00'
X'0063'	F00000	HALT2	HPL	X'00',X'00'
X'0066'	3000006B		SNS	STORE+1,X'00'
X'006A'	00000062	STORE	#	HALT1+2,#
X'006E'	0C0000640062		MVC	HALT2+1(1),HALT1+2
X'0074'	C0870060		B	HALT1

Figure 1-7. Local Storage Register (LSR) Display Routine

STORAGE REQUIREMENTS

The Absolute Card Loader requires the first 256 bytes of storage. The first 96 bytes are used as a read buffer (Figure 1-8).

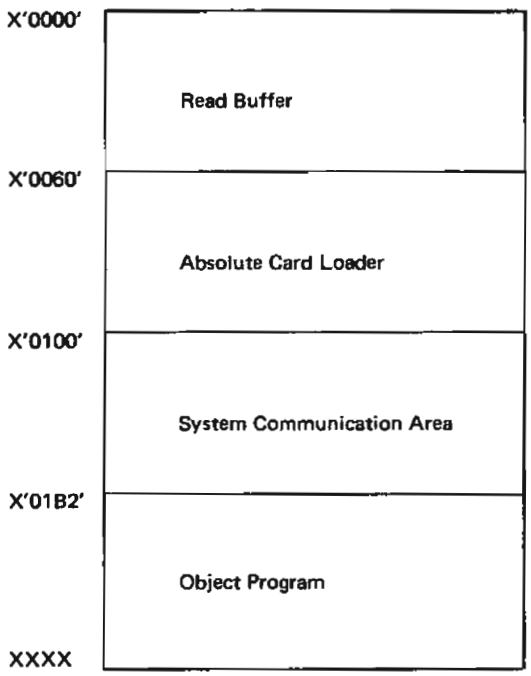


Figure 1-8. Storage Map for Absolute Card Loader

Figure 1-8 shows the location of the Absolute Card Loader in storage. See Chart AA for a flowchart of the Absolute Card Loader.

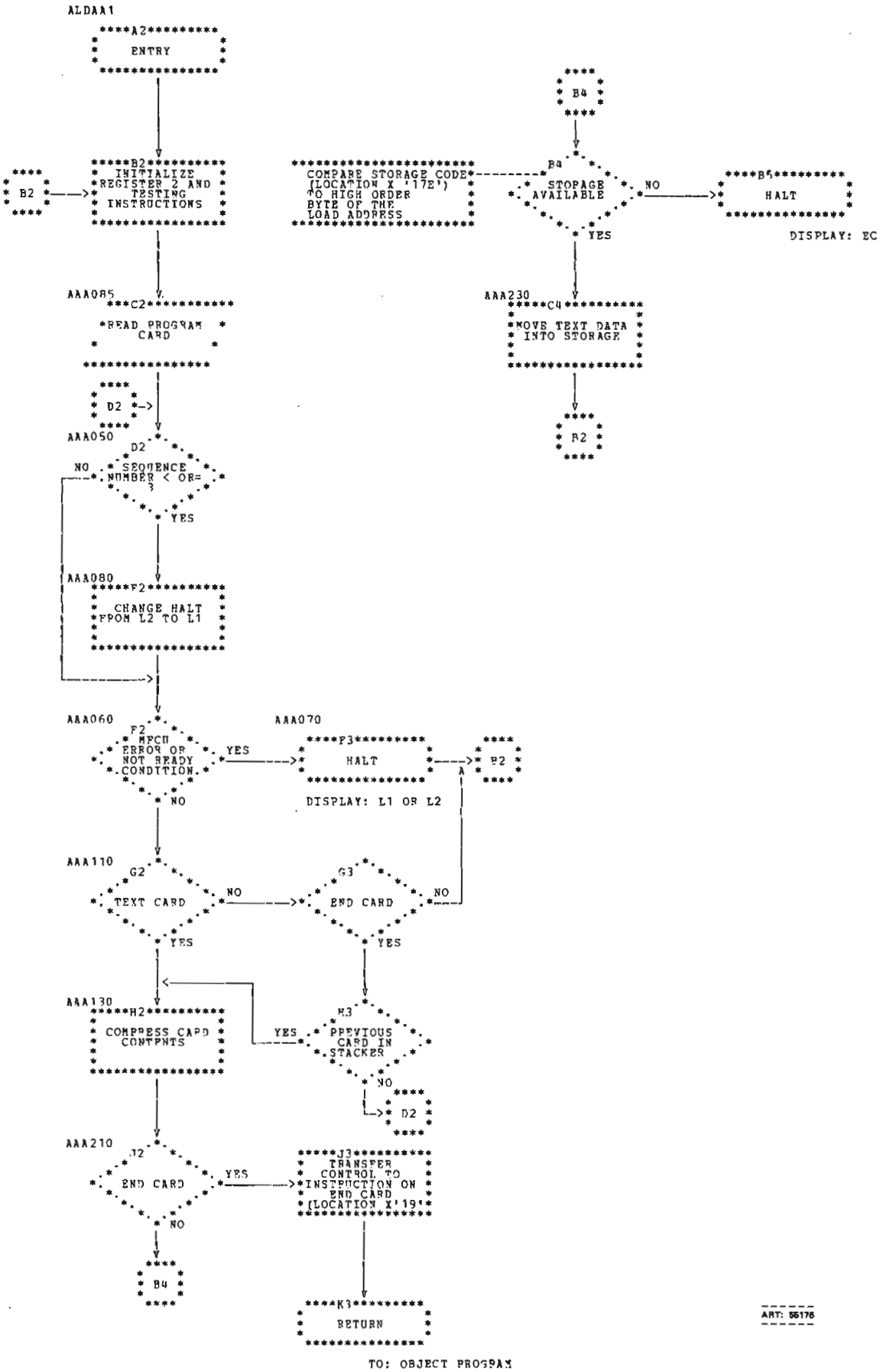


Chart AA. Absolute Card Loader

PART II
INPUT/OUTPUT CONTROL SYSTEM (IOCS)

Chapter 1. MFCU IOCS Routines

Section 1. Introduction

The IBM System/3 Card System Input/Output Control System (IOCS) for the IBM 5424 Multi-Function Card Unit (MFCU) is described in this chapter. The IOCS routines perform I/O (input/output) operations requested by IBM System/3 Card System programs. Error recovery procedures and operator restart procedures are provided for each IOCS routine. Halt identifiers, which appear on the Message Display Unit, and operator restart procedures are described in the *IBM System/3 Card System Operator's Guide*, Form C21-7513.

Figure 2-1 shows which IOCS routines are used by the IBM System/3 Card System programs. Figure 2-2 shows which IOCS routines are used by RPG II Compiler phases.

SYSTEM REQUIREMENTS

The IBM System/3 IOCS routines for MFCU use the following system configuration:

- IBM 5410 Processing Unit
- IBM 5424 Multi-Function Card Unit (MFCU)

Card System Programs \ MFCU IOCS Routines	Full Function (Object)	Full Function Modified (Object)	Read Only/Primary Hopper (Compiler)	Read Only/Primary Hopper (Object)	Read Primary/Punch Secondary (Compiler)	Read Only/Secondary Hopper (Compiler)	Read Only/Both Hoppers (Object)	Punch Only/Secondary Hopper (Compiler)
System Initialization	X							
Program Maintenance							X	
Device Counter Logout	X							
96 List				X				
96-96 Reproduce and Interpret	X							
Sort/Collate						X	X	
Data Recording		X						
Data Verifying		X						
80-96 Conversion	X							
RPG II Compiler			X		X	X		X
RPG II Compiler Object Program	X			X			X	

Figure 2-1. MFCU IOCS Routine Used by Card System Programs

Compiler Phase ID / Compiler MFCU IOCS Routine	RGAA	RGAC	RGAE	RGAG	RGAI	RGAK	RGCA	RGCC	RGCI	RGCK	RGEA
Read Only/Secondary Hopper	X	X	X	X	X	X	X		X	X	
Read Only/Primary Hopper								X			
Read Primary/Punch Secondary											X
Punch Only/Secondary Hopper										X	

Note 1: These compiler phases have no MFCU IOCS routines in storage:

- RGAM
- RGAO
- RGAP-RGBZ
- RGCE
- RGCG
- RGCJ
- RGCY-RGDC
- RGGZ
- RGJA
- RGJC
- RGJM

Note 2: All compiler phases not listed in this figure use the Punch Only/Secondary Hopper IOCS routine.

Figure 2-2. MFCU IOCS Routines Used by RPG II Compiler Phases

Section 2. Method of Operation

The two major functions of the MFCU IOCS are to (1) identify and perform the requested IOCS function and (2) detect and recover from any MFCU I/O errors. Figure 2-3 shows the functional and data flow for object program IOCS requests. Figure 2-4 shows the functional and data flow for compiler IOCS requests.

PERFORM IOCS FUNCTION

Object Program

The calling program branches to the MFCU IOCS routine to request an MFCU function. An IOB (input/output block) passed by the calling program specifies either a wait or an execute call.

A wait call performs these functions:

- Waits for the requested operation to complete by waiting until the buffer is not busy.
- Checks for errors and branches to the error recovery procedures if an error is detected.
- Clears the punch buffer to blanks if a punch call is requested.
- Determines which of the two print buffers is to be used if a print call is requested. The buffer is cleared to blanks and its address returned to the calling program in index register 2.

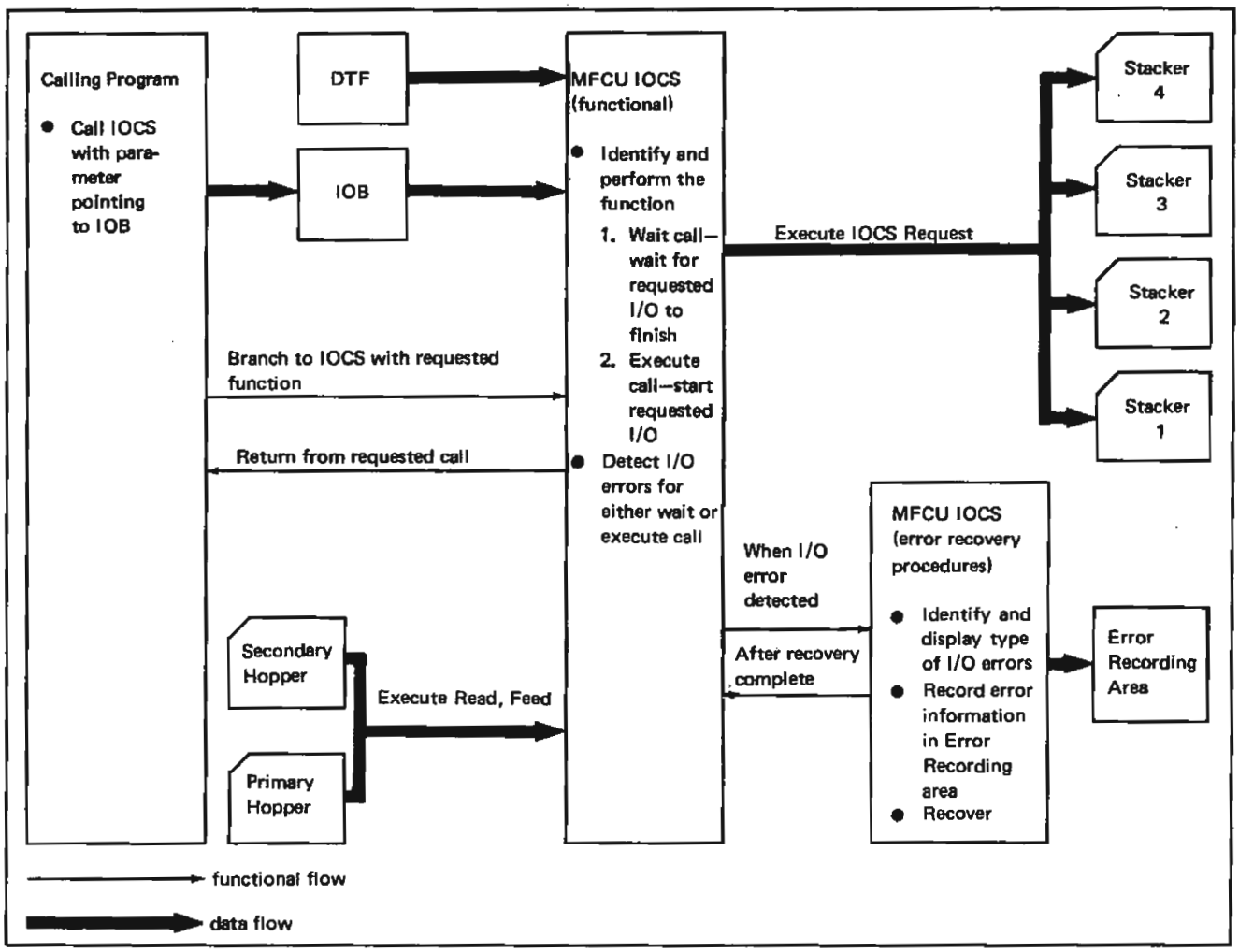


Figure 2-3. Functional and Data Flow for Object Program MFCU IOCS Requests

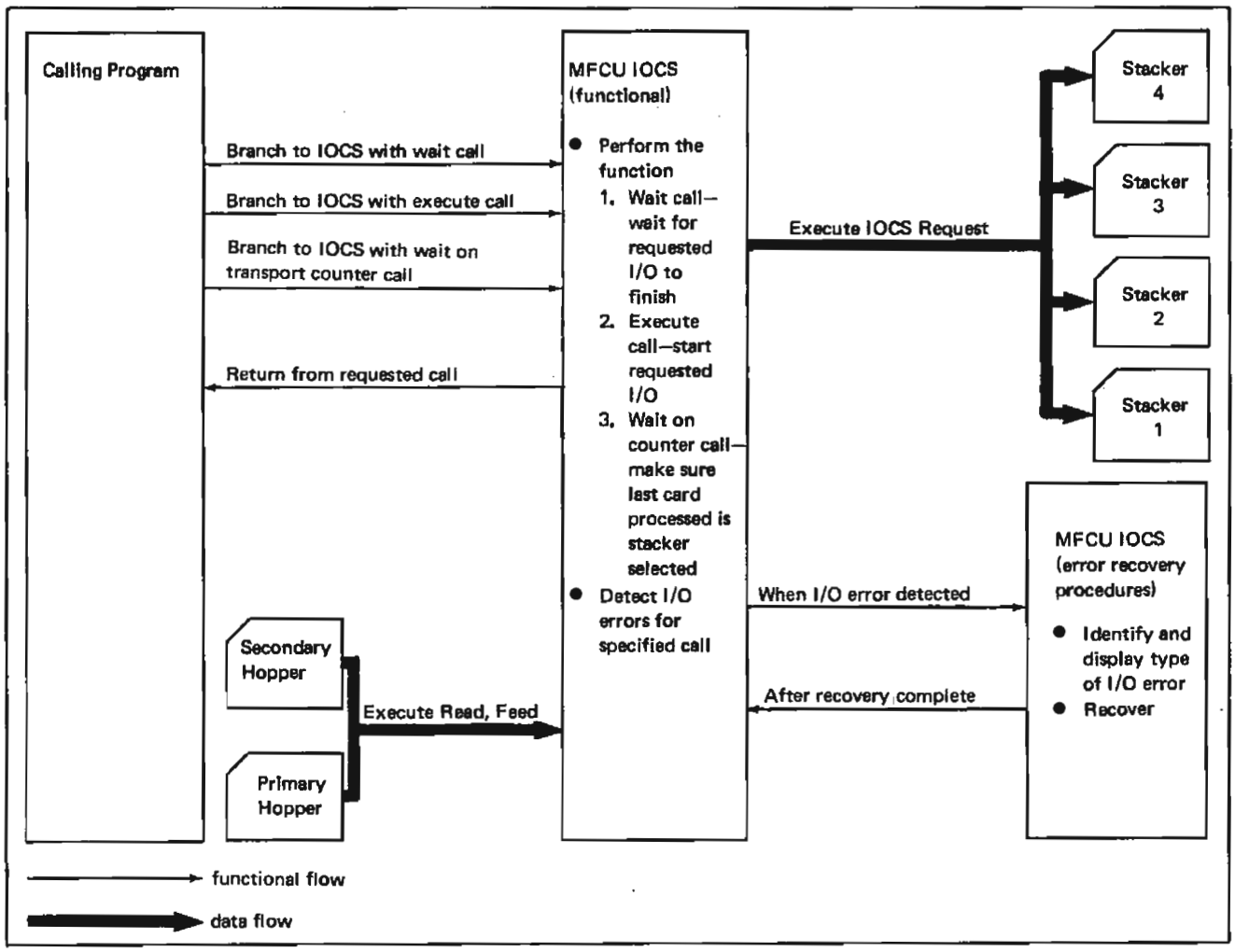


Figure 2-4. Functional and Data Flow for Compiler MFCU IOCS Requests

An execute call performs these functions:

- Checks for errors and branches to the error recovery procedures if an error is detected.
- Attempts to start the requested I/O operation.

Compiler

Compiler IOCS routines do not use an IOB or DTF (define the file). The entry point determines the function to be performed. These functions can be requested:

1. Execute call.
2. Wait call for buffer not busy.
3. Wait for all cards to clear transport.

DETECT AND RECOVER FROM I/O ERRORS

Object Program

Both the wait call and execute call check for I/O errors. If an error is detected, the IOCS routine branches to the error recovery procedures (ERPs). The ERPs attempt to recover from an I/O error by re-trying as many of the previous operations as necessary to properly complete the functions and stack the cards.

Error information is recorded in the Error Recording area for all errors except punch invalid and errors occurring during error recovery. A halt code is displayed when the error is detected. For a detailed description of the ERPs, see Section 3, *Program Organization.* Figure 2-6 is a summary of the halt codes for the error types and the program action taken.

The MFCU recovers from six types of errors:

- Feed check – The halt code is F1 for the primary hopper and F0 for the secondary hopper. A second halt displays the number of cards to be replaced in the hopper and the last selected stacker. After the cards have been replaced in the hopper, they are fed into the wait station and then restacked. If a print operation was not completed on the next to the last operation, it is repeated before the card is stacked. All operations not completed on the last operation are repeated before the card is stacked.
- Print check – The halt code is F6. No program error recovery is done. The program continues to check for errors.
- Punch invalid – The halt code is F5. A second halt displays the number of the stacker containing the card in error. No program error recovery can be done. The program continues to check for errors.
- Punch check – The halt code is F4. A second halt displays the number of the stacker containing the card in error. The punch operation is repeated.
- Read check – The halt code is F3. After the card is replaced in the hopper, it is reread. Punch and print operations are not repeated because they have been completed.
- Hopper check – The halt code is F2. After the error is corrected, the card is read or fed. Punch and print operations are not repeated because they have been completed.

Compiler

Compiler IOCS ERPs differ from object program IOCS ERPs in that they do not record error information in the Error Recording area.

Section 3. Program Organization

GENERAL DESCRIPTION OF MFCU IOCS ROUTINES

The overview presented in this section deals with an object program IOCS routine. Compiler IOCS routines differ from object program IOCS routines in these ways:

- Compiler IOCS routines do not use an IOB or DTF.
- Compiler IOCS routines have separate entry points. The possible entry points are:
  1. Execute call.
  2. Wait call.
  3. Wait for all cards to clear transport.
- Compiler IOCS routines do not record error information in the Error Recording area.
- Compiler IOCS routines do not clear the punch buffer if a wait call has been requested.

See *MFCU IOCS Routines* in this section for flowcharts of specific routines.

Functional Description

The calling program branches to the MFCU IOCS routine to request an MFCU function. Immediately following the branch is a 2-byte parameter which points to the IOB. The IOB contains the type of call, the function to be performed, the record length, and stacker information.

The last two bytes of the IOB point to the 14-byte DTF. The DTF contains buffer addresses needed by the IOCS routines and provides space for storage of error information (see Section 4, *Data Area Formats,* for the contents of the IOB and DTF). Figure 2-5 shows the linkage of these areas.

After initialization is done, IOCS checks the IOB to determine whether a wait call or an execute call is requested.

Wait Call

If a wait call is requested, IOCS performs the following steps to complete the functions:

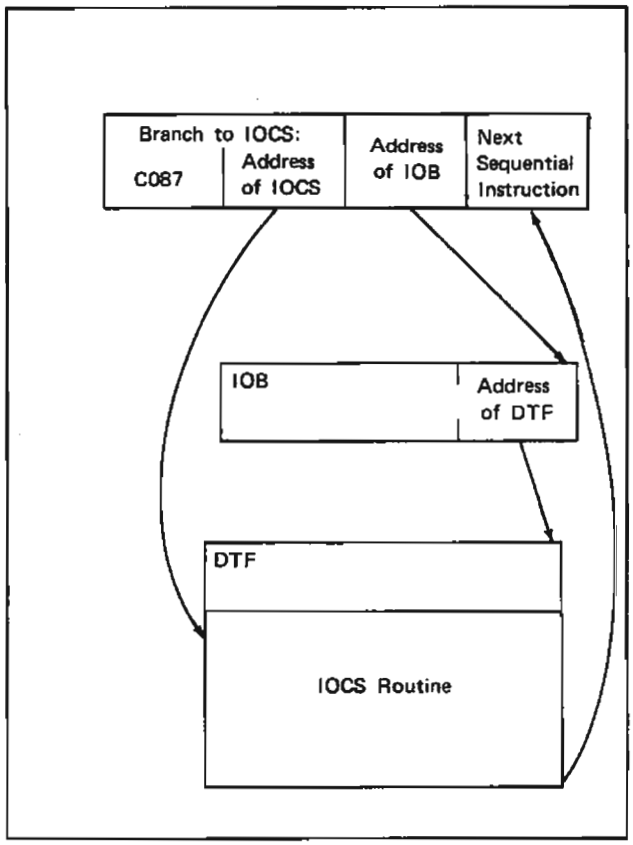


Figure 2-5. Linkage for Object Program MFCU IOCS Routines

- Builds an APL instruction to wait for the specified buffer to be not busy.
- Checks for device errors and branches to ERPs if an error is detected.
- Moves blanks into the punch buffer specified in the DTF if a punch call is requested.
- Determines if a print call is requested and, if so, does the following:
  1. Assigns one of the two print buffers to be used in the next execute call.
  2. Stores the address of the assigned buffer in the DTF.
  3. Loads the address in index register 2.
  4. Clears the buffer to blanks.

IOCS then returns to the calling program. Chart IA, part 1, shows an overview of the wait call. For a more detailed flowchart, see *Detailed Description of the MFCU IOCS Routines* in this section.

Execute Call

If an execute call is requested, IOCS performs the following steps to complete the functions:

- Waits for all previous I/O operations on the MFCU to complete.
- Checks for device errors and branches to ERPs if an error is detected.
- Updates the History Table with stacker information from the R byte of the last SIO instruction (see Section 4, *Data Area Formats*, for the contents of the History Table).
- Builds the SIO instruction by moving in the Q byte from the APL instruction in the wait call and the R byte from its build area.
- Sets up feed check error recovery for the specified hopper.
- Initiates the I/O operation (SIO).

When the I/O operation has been completed for a wait call or accepted for an execute call, IOCS returns to the calling program. Chart IA, part 1, shows an overview of the execute call. For a more detailed flowchart, see *Detailed Description of the MFCU IOCS Routines* in this section.

Error Recovery Procedures Description

The wait call or execute call branches to the error recovery procedures when it detects a device error. The status bytes are sensed into the DTF and loaded into the ARR (address recall register) from the DTF. Figure 2-6 shows the status sense information for each error.

A check is then made for I/O errors. If none have occurred, the error is a simple device-not-ready; the IOCS ERP returns to the IOCS mainline routine.

Halt Code	Condition	Priority	Status		I/O Attention Light	Program Action
			Byte	Bit		
None	Simple not ready	4	—	—	On	Loop on the SIO until it is accepted.
None	No-op	4	1	7	Off	Prime the wait station.
F0 or F1	Feed check	1	1	6	Off	Reissue all operations not completed.
F6	Print check (see Note 3)	2	1	3,4	Off	Continue processing.
F5	Punch invalid	2	1	2	Off	Continue processing.
F4	Punch check	2	1	1	Off	Reissue the last command. If read was specified, set off read bit before reissuing command.
F3	Read check	3	1	0	Off	Reissue read portion of last command.
F2	Hopper check	3	1	5	Off	Feed a card or, if the last command was a read, read a card.

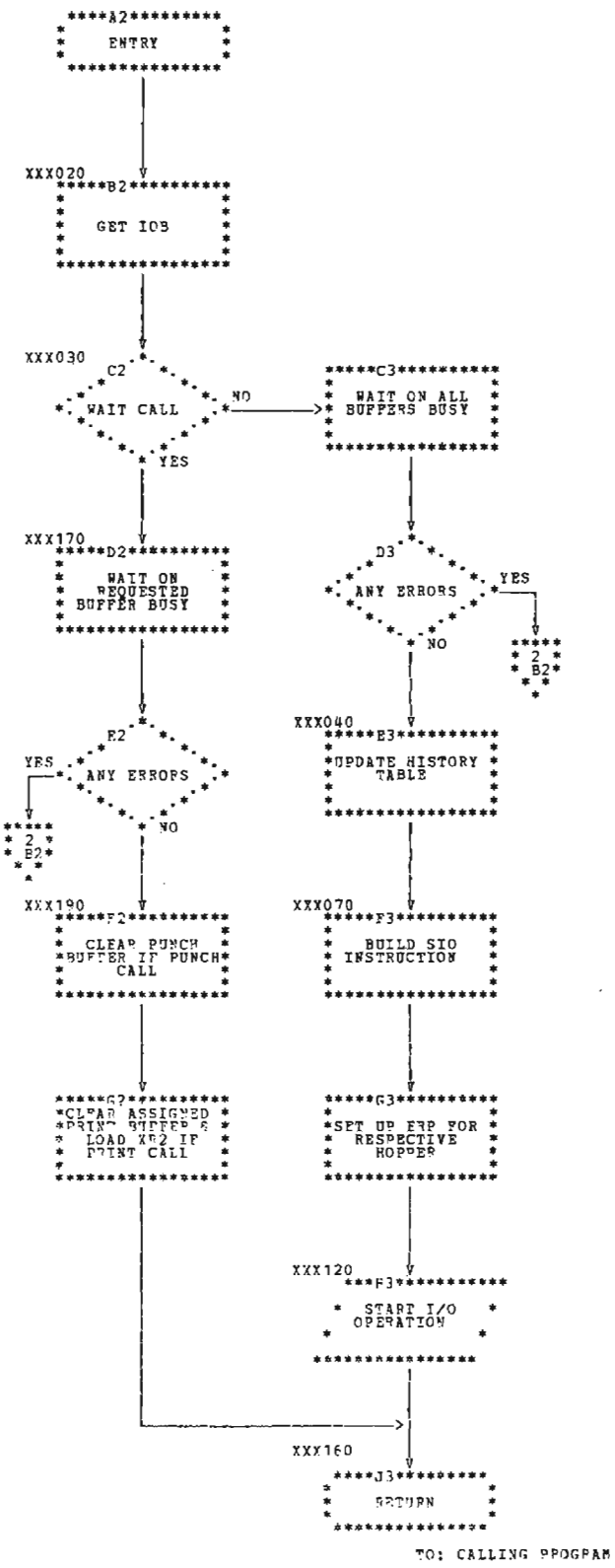
Note 1: Errors are processed in the order they appear on this chart.

Note 2: Byte 1 refers to the second byte (high storage address) of sense data.

Note 3: Two types of print check can occur:

1. Print data check.
2. Print clutch check.

Figure 2-6. Error Recovery Information for MFCU IOCS Routines



Note: The labels on this chart correspond to the labels on the chart for the Full Function routine.

Chart IA. Overview–Wait and Execute Calls (Part 1 of 2)

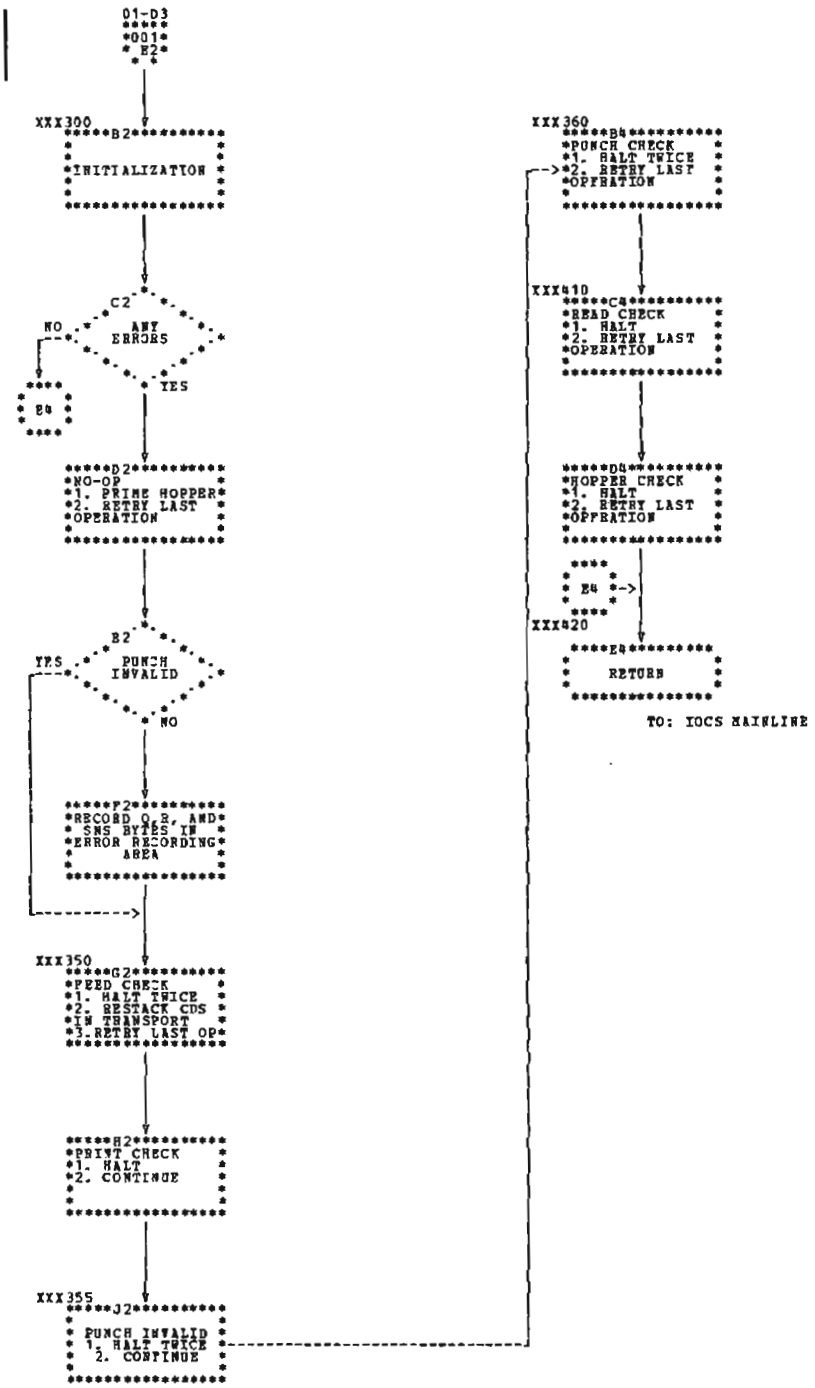


Chart IA. Overview–Error Recovery Procedures (Part 2 of 2)

If an I/O error has occurred, a test is made to see if the no-op bit is on. This bit, if on, indicates that the wait station must be primed before the hardware can perform the required punch or print operation. The wait station is primed and the program returns to the IOCS mainline to build the SIO instruction.

If the no-op bit is off, six types of errors are checked for in this order:

1. Feed check
2. Print check
3. Punch invalid
4. Punch check
5. Read check
6. Hopper check

The Q, R, and SNS bytes in the DTF are moved into the Error Recording area for all errors except punch invalid and errors occurring during error recovery. See Section 4, *Data Area Formats*, for a description of the Error Recording area.

Chart IA, part 2, shows an overview of the ERP. For a more detailed flowchart, see *Detailed Description of the MFCU IOCS Routines* in this section.

Feed Check

The halt code for a feed check is displayed (F0 or F1). The program maintains two counters. The display counter keeps track of the number of cards in the transport, including the wait station. The command counter keeps track of the number of commands which need to be repeated. The program builds the second halt by referencing the History Table to determine the last selected stacker and the display counter to determine the number of cards to be replaced in the hopper. The correct codes for the second halt are then selected from the Halt Table and this halt is displayed.

The command counter is tested to see if all feeds have been completed. If all feeds are not complete, another card is fed into the wait station. The command counter is tested again. When all cards have been fed, the program returns to the IOCS mainline to repeat the last operation. Figure 2-7 shows a schematic diagram of feed check ERP.

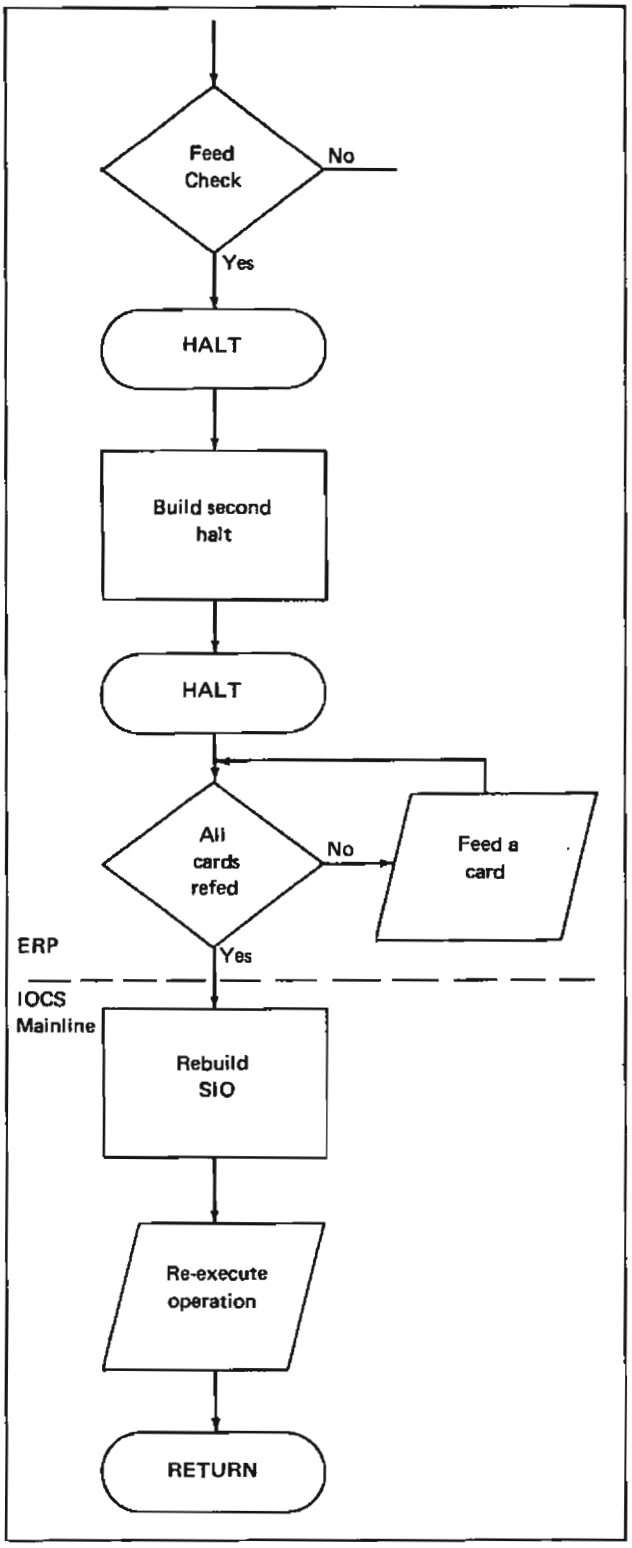


Figure 2-7. Schematic Diagram of Feed Check ERP

Print Check

The halt code for a print check is displayed (F6). Processing continues to determine if there are other errors. Figure 2-8 shows a schematic diagram of print check ERP.

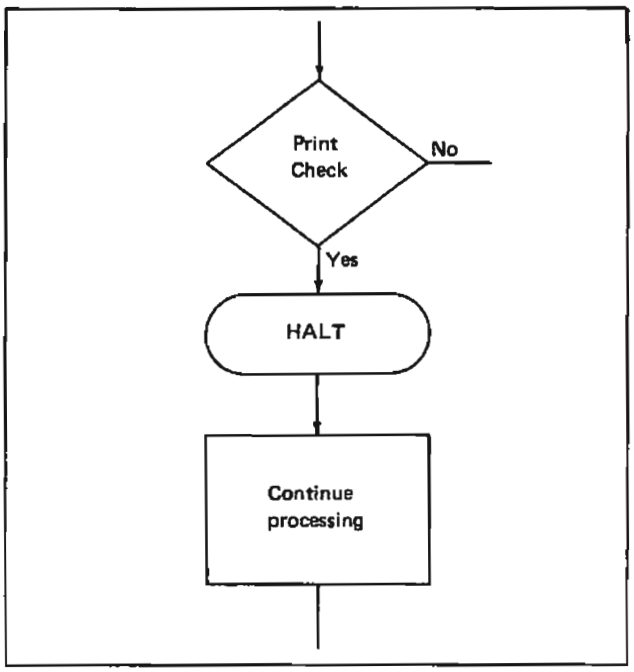


Figure 2-8. Schematic Diagram of Print Check ERP

Punch Invalid

The halt code for a punch invalid is displayed (F5). The program references the History Table to determine the stacker which contains the incorrect card and selects the correct halt code from the Halt Table. This second halt is displayed. Processing continues to determine if there are other errors. Figure 2-9 shows a schematic diagram of the punch invalid ERP.

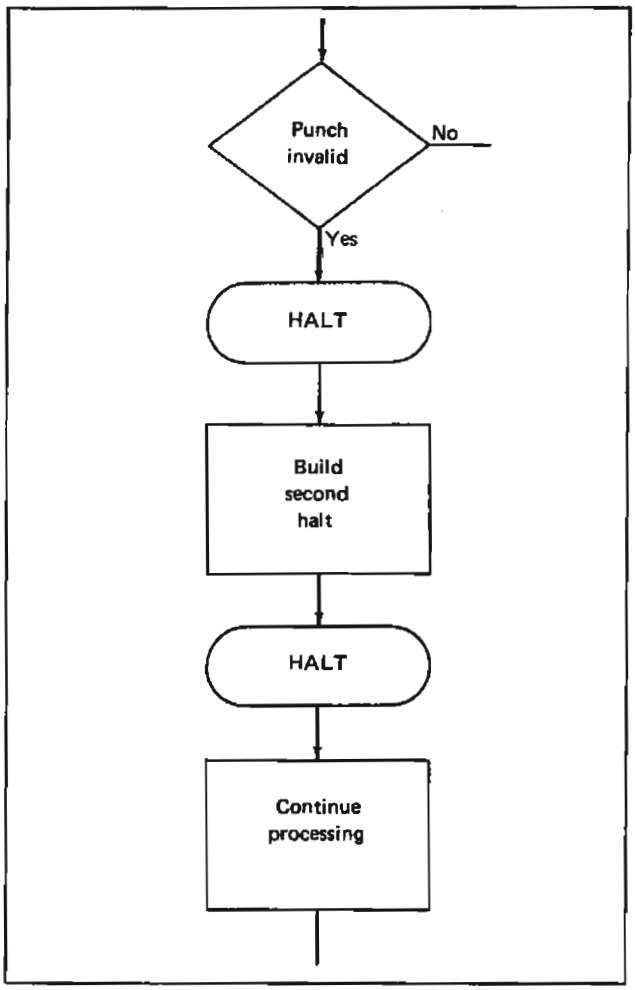


Figure 2-9. Schematic Diagram of Punch Invalid ERP

Punch Check

The halt code for a punch check is displayed (F4). The program references the History Table to determine the stacker which contains the incorrect card and selects the correct halt code from the Halt Table. This second halt is displayed. The program returns to the IOCS mainline to repeat all but the read portion of the last operation. Figure 2-10 is a schematic diagram of the punch check ERP.

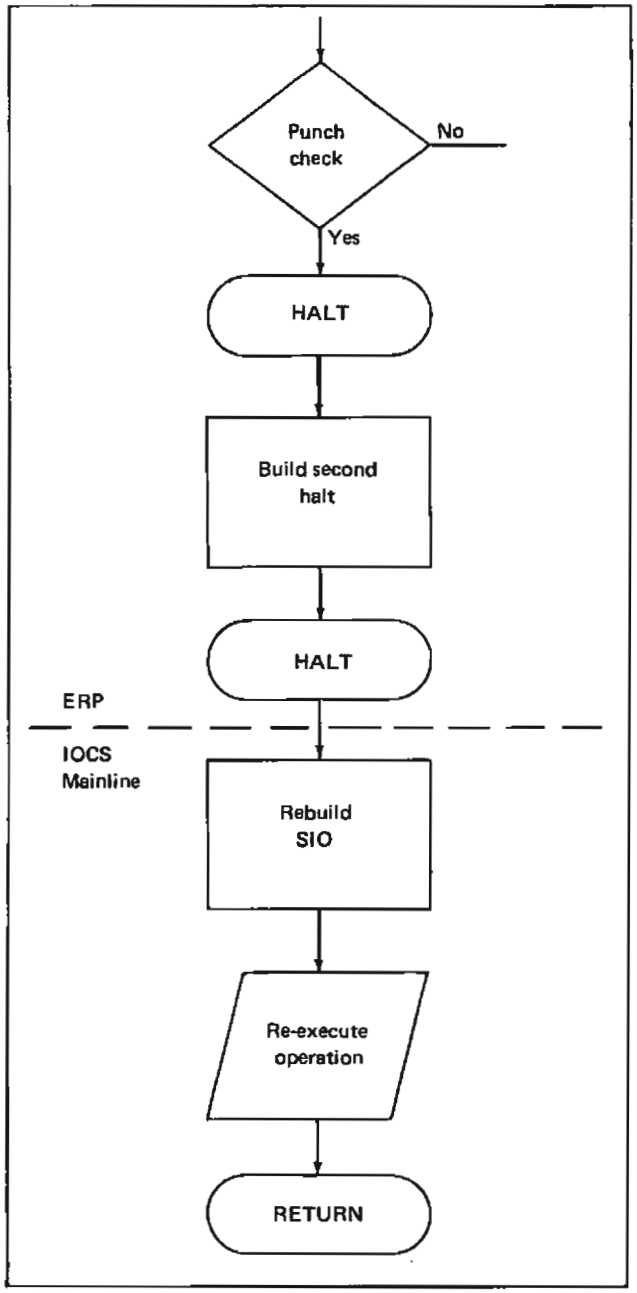


Figure 2-10. Schematic Diagram of Punch Check ERP

Read Check and Hopper Check

Error recovery is the same for these two errors. The correct halt code is displayed (F2 or F3). The program returns to the IOCS mainline to repeat the read or feed portion of the last operation. Figure 2-11 is a schematic diagram of the read check and hopper check ERP.

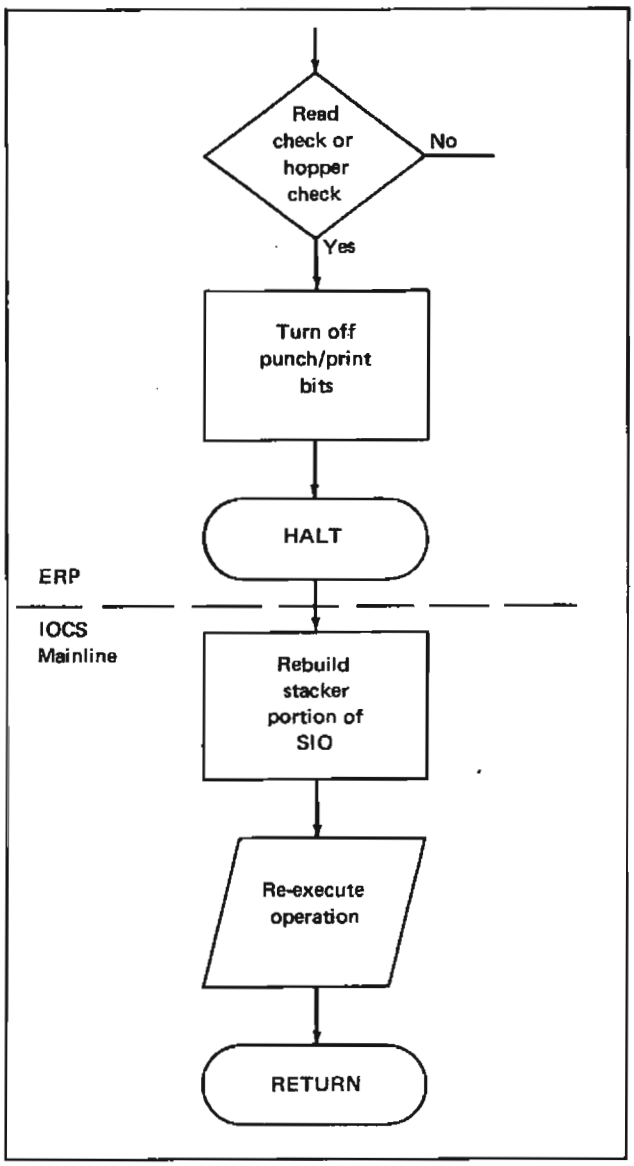


Figure 2-11. Schematic Diagram of Read Check or Hopper Check ERP

DETAILED DESCRIPTION OF THE MFCU IOCS ROUTINES

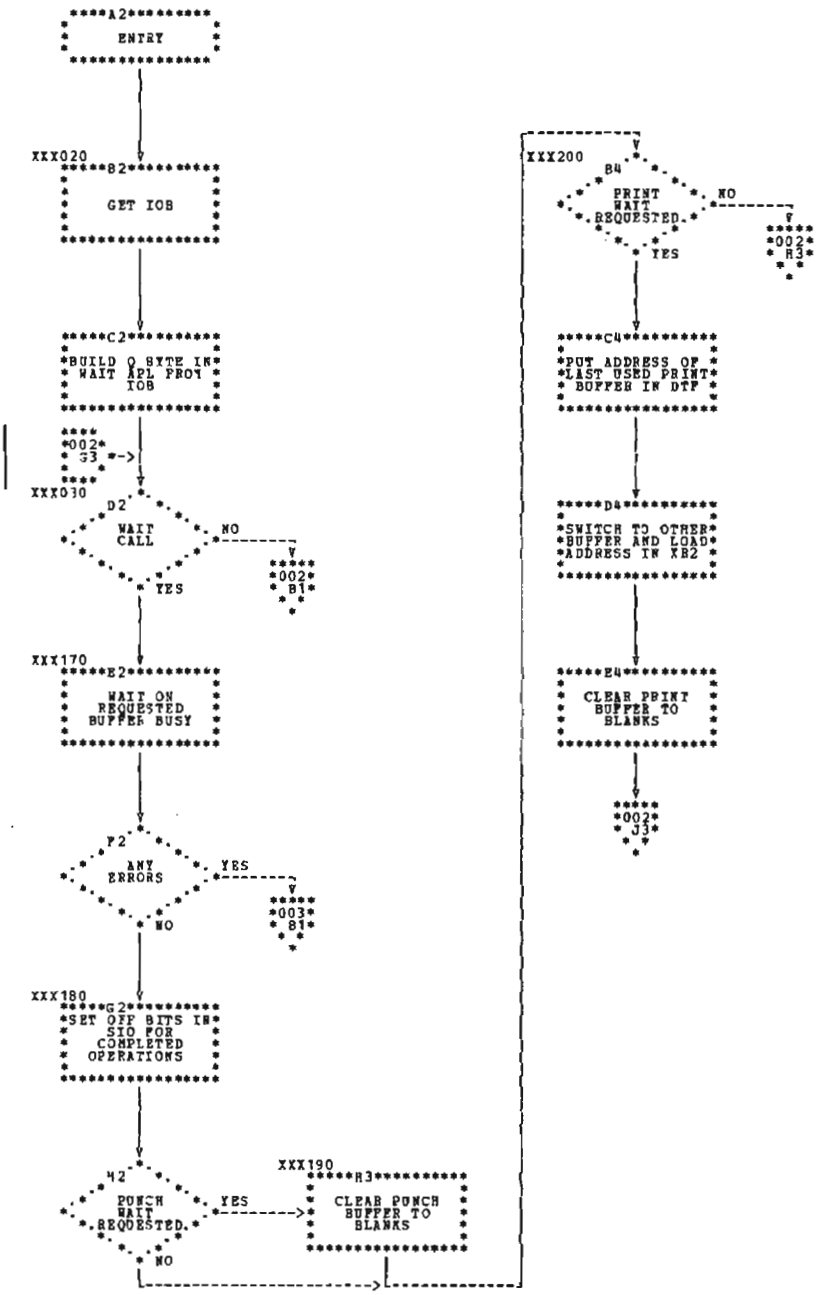
This section contains detailed flowcharts of the overview flowchart presented previously (Chart IA). Chart IB is broken down in this manner:

- Part 1 – Initialization and wait call
- Part 2 – Execute call
- Parts 3 and 4 – Error recovery procedures

Notes relating to specific blocks in the flowcharts are included for more thorough explanations.

01G2: Bits are set off so completed operations will not be repeated. Only uncompleted operations and stacker selection need be performed.

01C4-01E4: Two print buffers are used alternately for maximum throughput. When a print wait is requested, the buffer address is switched in the DTF and loaded in XR2. The new address in the DTF will be used by the next print operation.



Note: The labels on this chart correspond to the labels on the chart for the Full Function routine.

Chart IB. Initialization and Wait Call (Part 1 of 4)

02D1: Save the print bit from the Q byte of the SIO of the last operation. This bit is used to condition print re-try during error recovery.

02F1: Stacker information is obtained from the IOB. The print-4-lines bit is derived from the record length in the IOB. The print buffer bit is derived from the print buffer address in the DTF.

02C3, 02B5: Error recovery depends on the hopper specified in the current operation.

1. ERP is set up to refill the opposite wait station since the entire transport must be cleared when a feed check error occurs. If no operation from the opposite hopper has been processed, this step is bypassed.

2. The test for a card in the wait station is initialized. If a card is in the wait station, ERP increments the command and display counters by one.

02E3: This decision is controlled by block 04C3.

02C5: If a punch or print is not requested on the first operation from the secondary hopper, no card is stacked and the History Table is not updated.

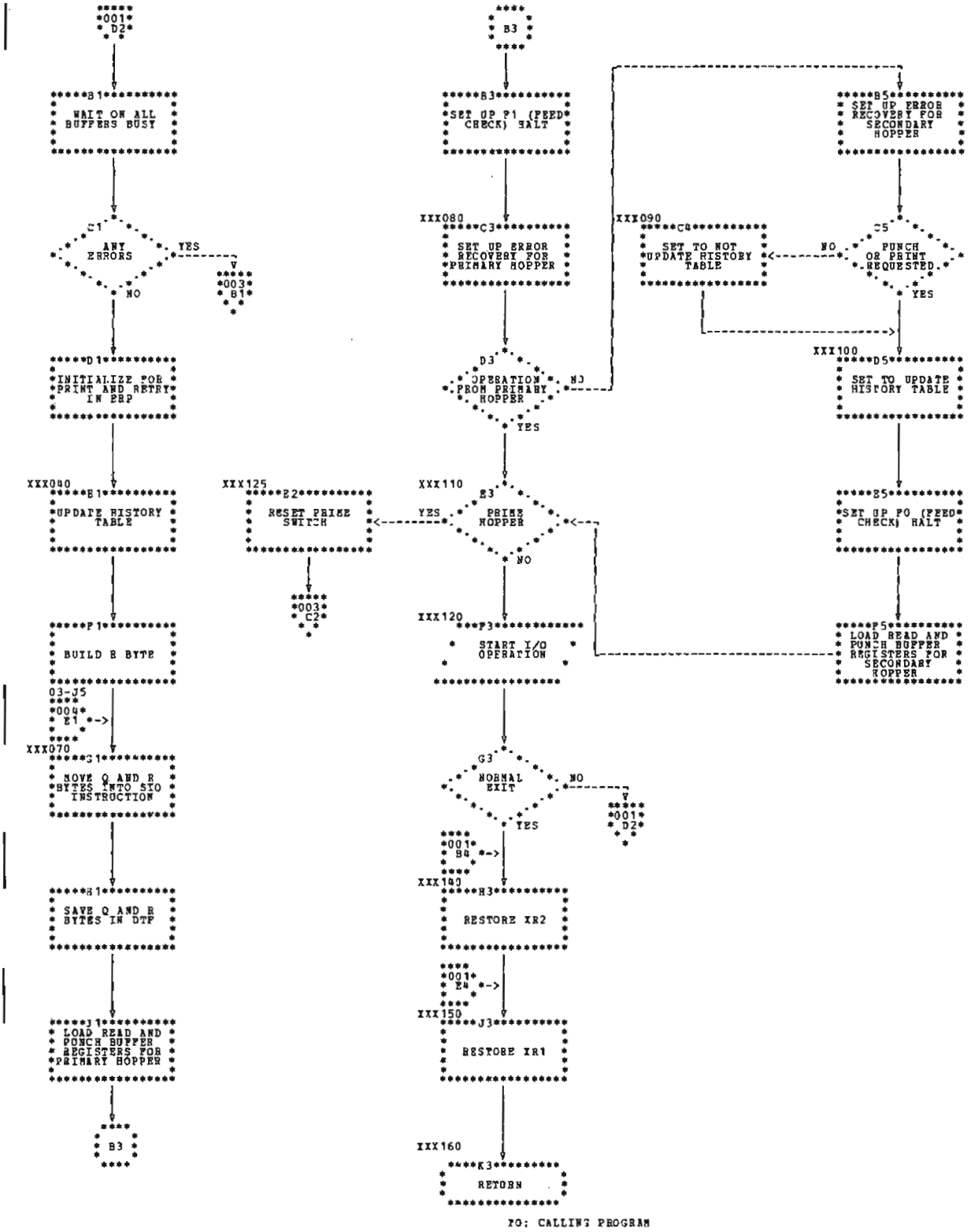


Chart IB. Execute Call (Part 2 of 4)

03C1-03D1: The program waits for the hopper-cycle-not-complete bit to be static.

03F1: If there are no errors, a simple device-not-ready has occurred and does not require error recovery. Device-not-ready can be caused by:

1. Hopper empty.
2. Stacker full.
3. Chip box full.
4. Covers open.
5. MFCU STOP key pressed.

03F1-03H1: The program waits for all card motion to stop. A feed check causes the machine to stop immediately which may leave cards in the transport. Other errors allow completion of the operation. The check for transport counter = 0 waits for the operation, including stacking, to be completed.

03J1: If an error has occurred during feed check error recovery, error recording is bypassed.

03C3: A no-op condition indicates that the first command for the hopper includes a punch or print but there is no card in the wait station. No error has occurred but the wait station must be primed before the operation can be executed.

03G5: These bits are set off because these operations have already been completed and must not be repeated.

03J5: The 'no' exit will be taken only if the error occurs during the feeds of feed check error recovery. If an error occurs during one of these operations, the feed must be re-tried without decrementing the command counter.

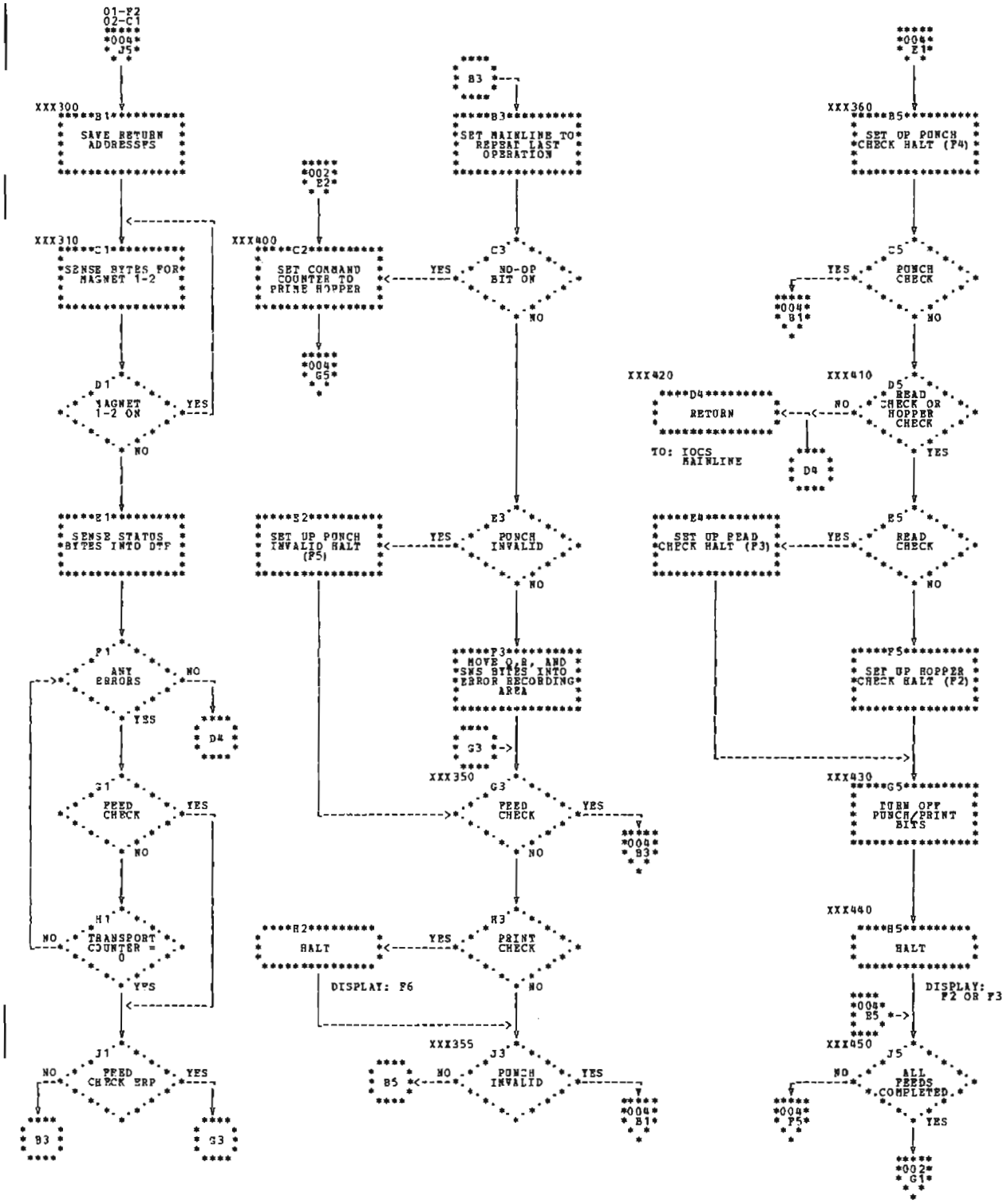


Chart IB. Error Recovery Procedures (Part 3 of 4)

04B3-04J3: The display counter is equal to the number of cards in the transport past the wait station plus one if a card is in the wait station.

The command counter is equal to the number of cards calculated for the display counter plus one if a card should have left the hopper but has not.

The number of cards in the transport past the wait station is indicated by the hardware transport counter. If a card is in the wait station, the card-in-wait bit is on or both the card-in-wait bit and the hopper-cycle-not-complete bit are off.

04C3: If a card was in the opposite wait station, it must be fed into the wait station because the transport is cleared during feed check error recovery.

04F5: Printing may not have been completed so the print operation must be repeated.

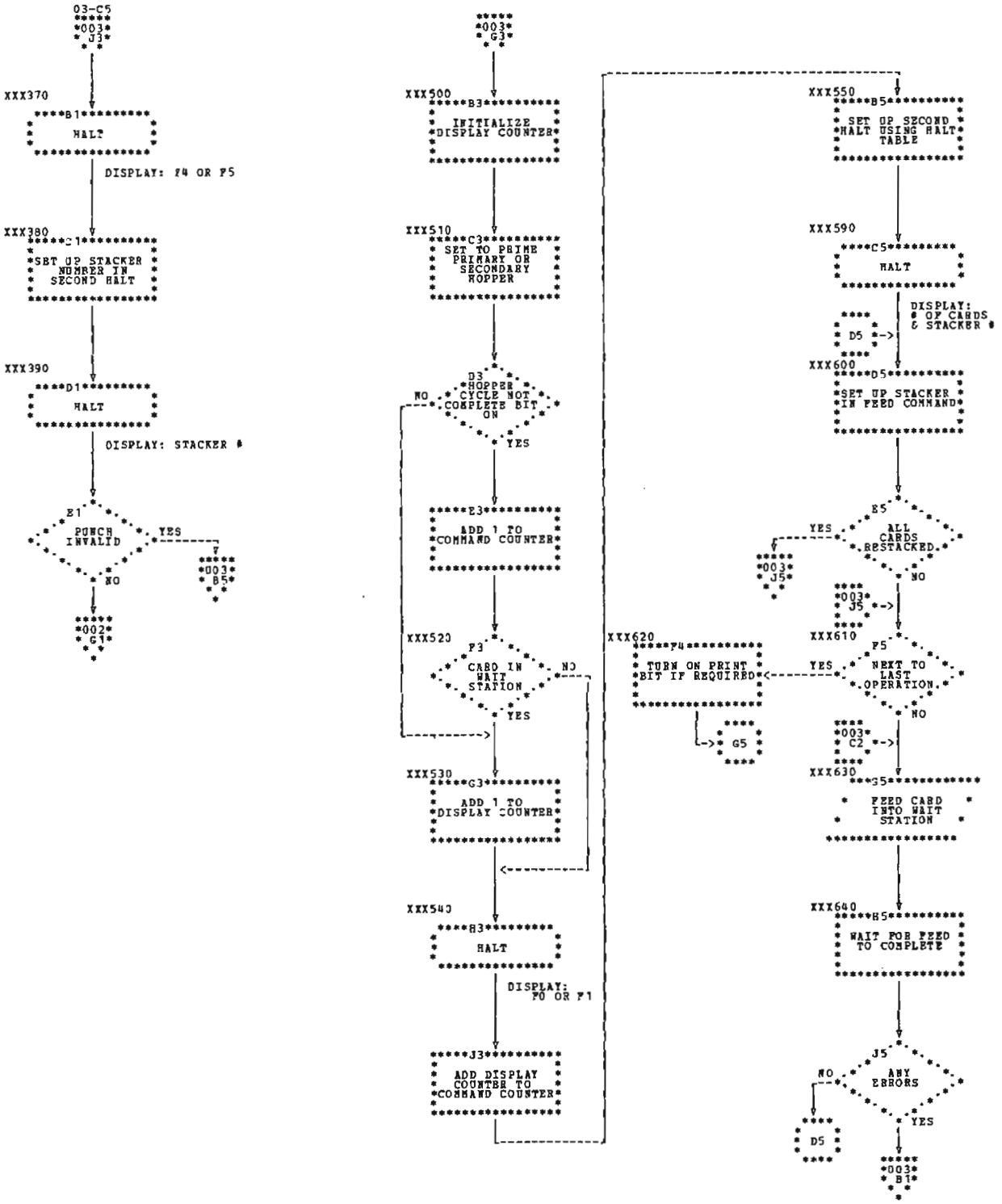


Chart IB. Error Recovery Procedures (Part 4 of 4)

MFCU IOCS ROUTINES

This section gives a detailed description of each MFCU IOCS routine. The functions of each routine are given as well as its entry points, input required to perform the functions, resulting output, and exits. Distinctions from the general IOCS routine previously described are indicated. The flowcharts for a routine are on a level similar to Chart IA in *General Description of MFCU IOCS Routines.* For a more detailed flowchart and discussion, see Chart IB.

Full Function (Object)

Entry Point: ARGEM1

Chart: IC

Functions:

- Waits for the I/O operation to complete.
- Reads or feeds cards from the primary or secondary hopper.
- Punches and/or prints cards from either MFCU hopper.
- Provides stacker selection.
- Detects and recovers from errors.

Input:

- IOB
- DTF

Output:

- Address of the print buffer if print wait is specified.
- Blanks in punch/print buffers if punch/print is specified.
- Punched and/or printed output.

Exit: Control is returned to the instruction immediately following the 2-byte parameter pointing to the IOB.

Routine Distinctions: None

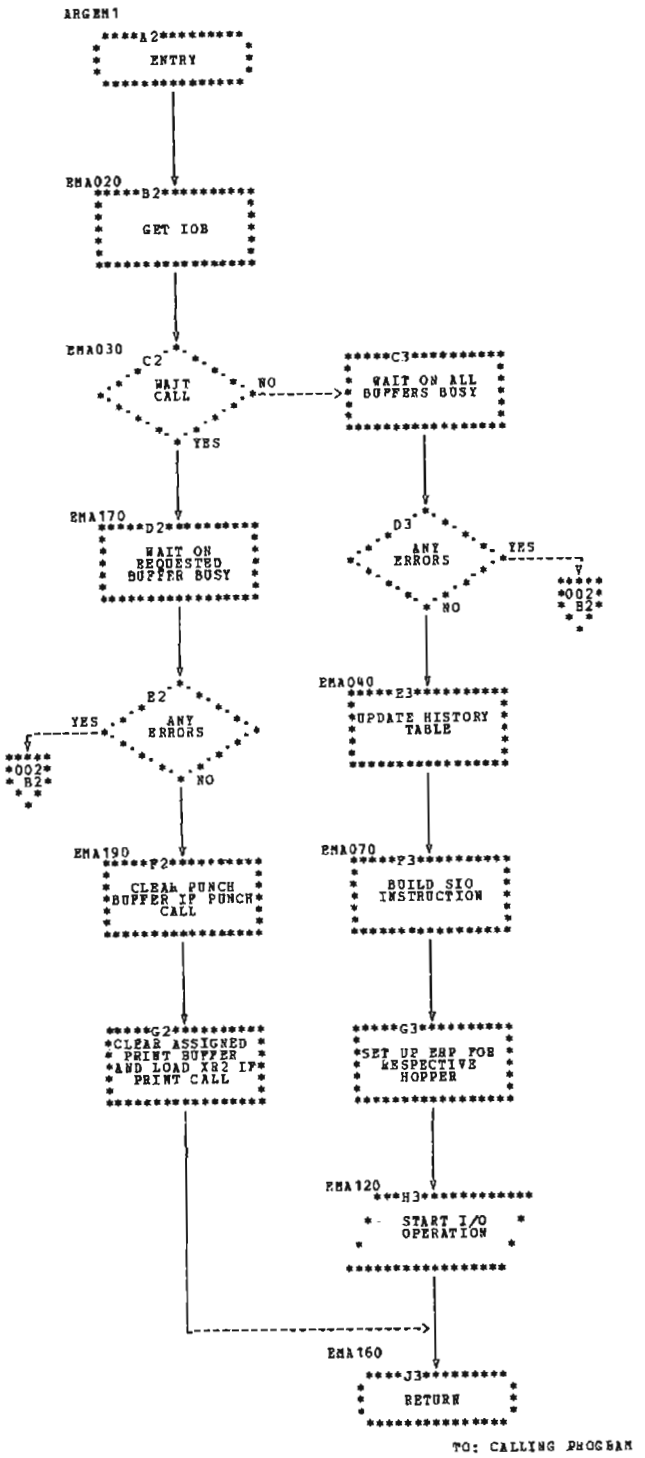


Chart IC. Full Function (Object)—Wait and Execute Calls (Part 1 of 2)

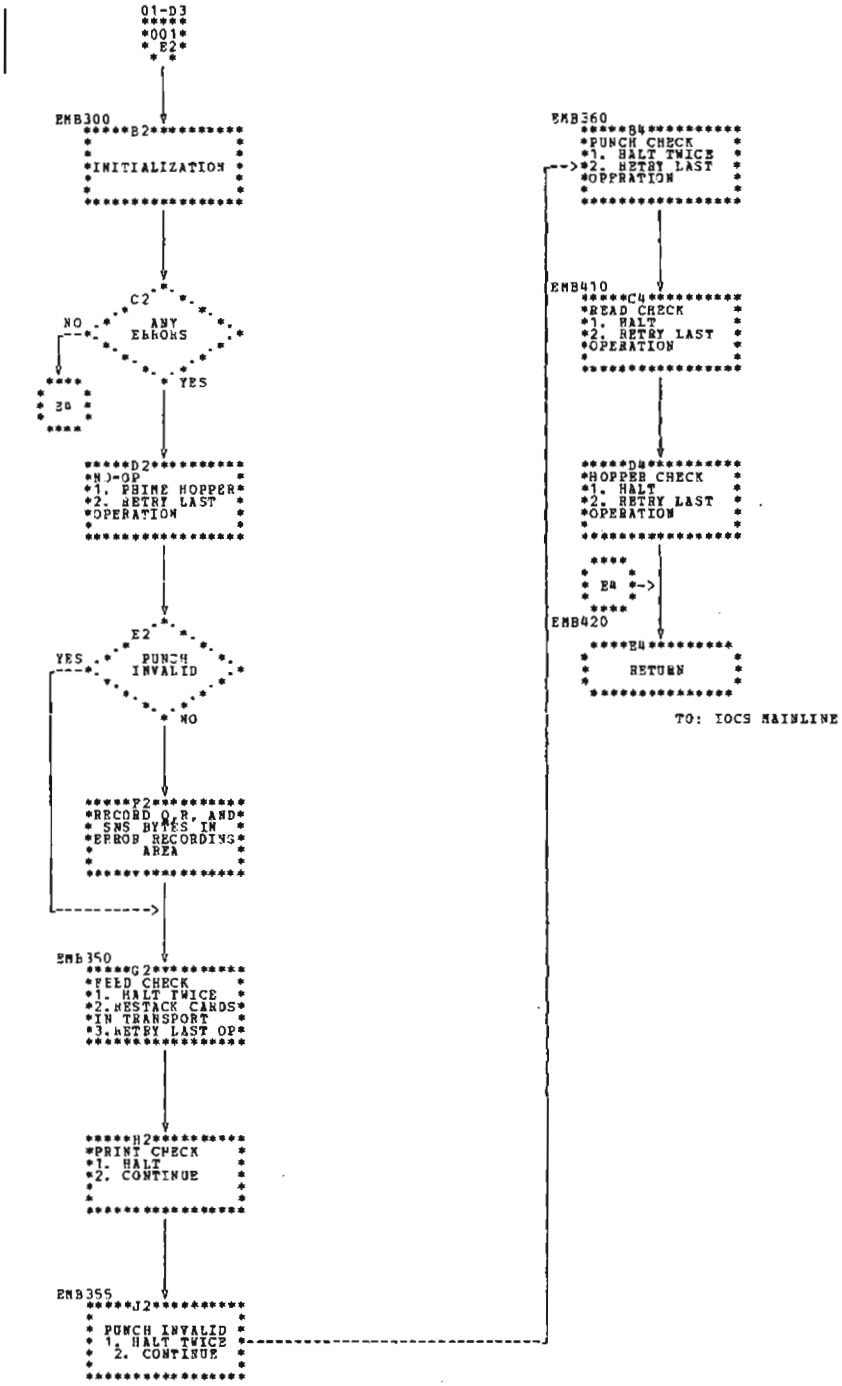


Chart IC. Full Function (Object)—Error Recovery Procedures (Part 2 of 2)

Full Function Modified for Data Recording/Verifying (Object)

Entry Point: ARGEN1

Chart: ID

Functions:

- Waits for the I/O operation to complete.
- Reads or feeds cards from the primary or secondary hopper.
- Punches and/or prints cards from either MFCU hopper.
- Provides stacker selection.
- Detects and recovers from errors.

Input:

- IOB
- DTF

Output:

- Address of the print buffer if print wait is specified.
- Blanks in punch/print buffers if punch/print is specified.
- Punched and/or printed output.

Exit: Control is returned to the instruction immediately following the 2-byte parameter pointing to the IOB.

Routine Distinctions:

- Only the second print buffer is used.
- Address of the print buffer is not put in index register 2.

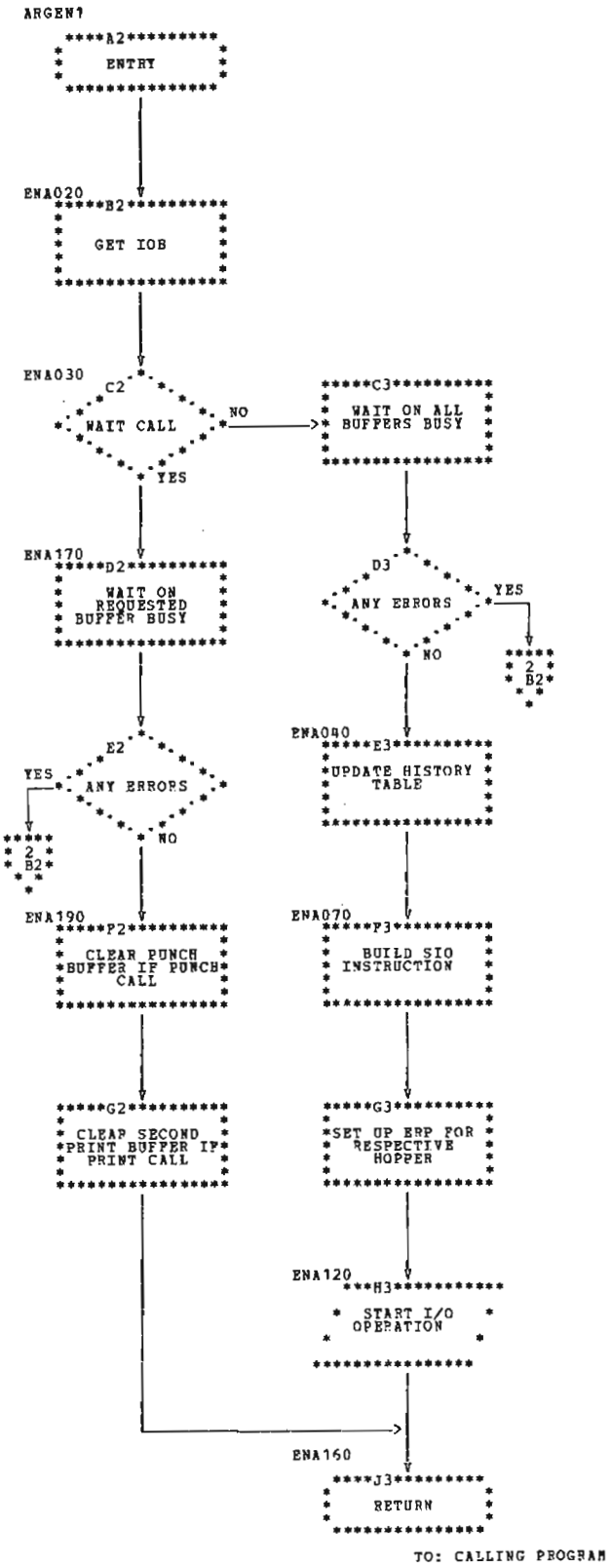


Chart ID. Full Function Modified (Object)–Wait and Execute Calls (Part 1 of 2)

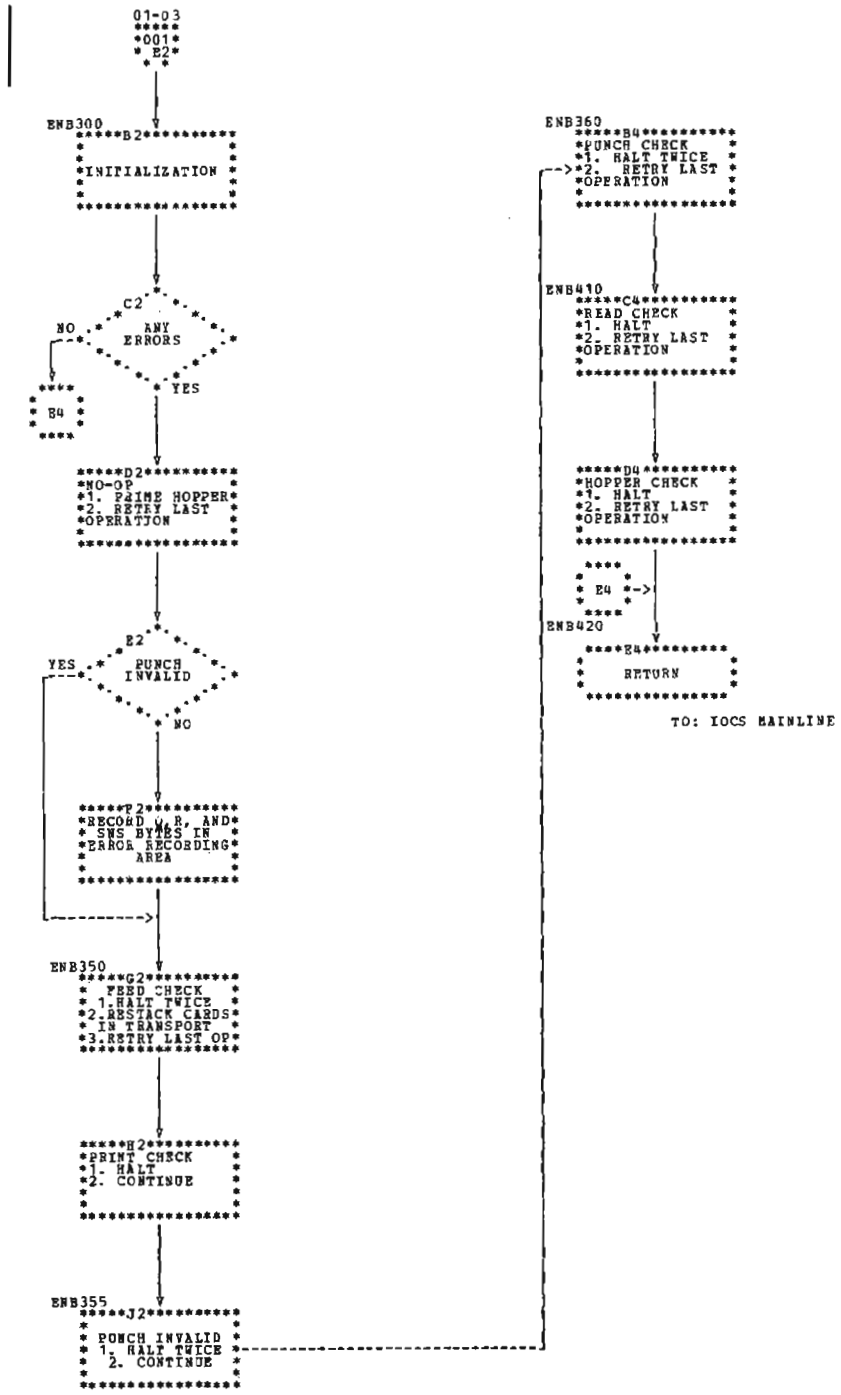


Chart ID. Full Function Modified (Object)—Error Recovery Procedures (Part 2 of 2)

Read Only/Primary Hopper (Compiler)

Entry Points:

- ARGEI1 – Execute call
- ARGEI2 – Wait call

Chart: IE

Functions:

- Waits for the I/O operation to complete.
- Reads cards from the primary hopper.
- Directs cards to stacker 1.
- Detects and recovers from errors.

Input: None

Output: None

Exit: Control is returned to the instruction immediately following the branch to this routine.

Routine Distinctions: All ERPs are provided except:

1. Print check.
2. Punch invalid.
3. Punch check.

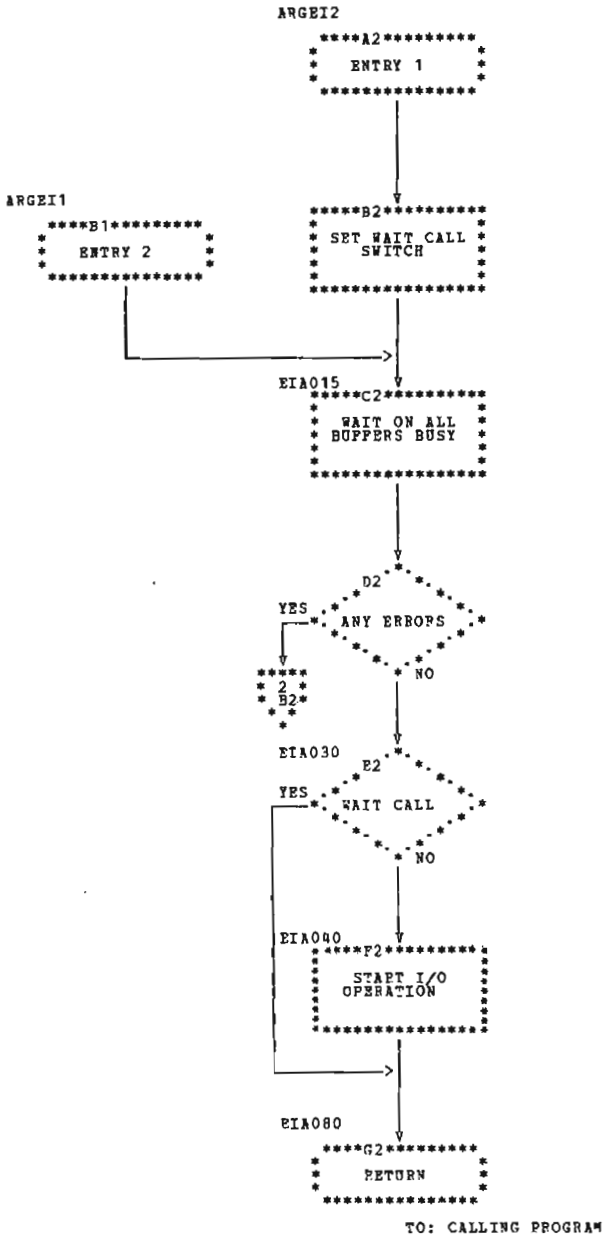


Chart IE. Read Primary (Compiler) – Wait and Execute Calls

(Part 1 of 2)

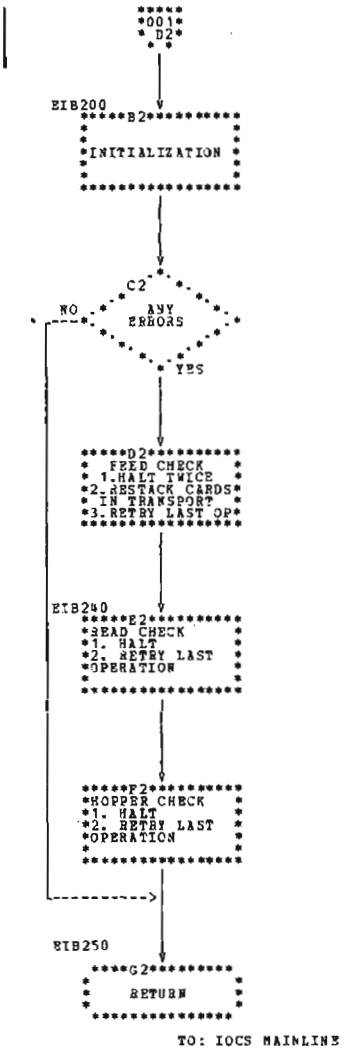


Chart IE. Read Primary (Compiler) – Error Recovery Procedures (Part 2 of 2)

Read Only/Primary Hopper (Object)

Entry Point: ARGEC1

Chart: IF

Functions:

- Waits for the I/O operation to complete.
- Reads cards from the primary hopper.
- Provides stacker selection.
- Detects and recovers from errors.

Input:

- IOB
- DTF

Output: None

Exit: Control is returned to the instruction immediately following the 2-byte parameter pointing to the IOB.

Routine Distinctions:

- Only the execute/wait bit and the read primary bit are interrogated. Any other bits that are on are ignored.
- All ERPs are provided except:
  1. Print check.
  2. Punch invalid.
  3. Punch check.

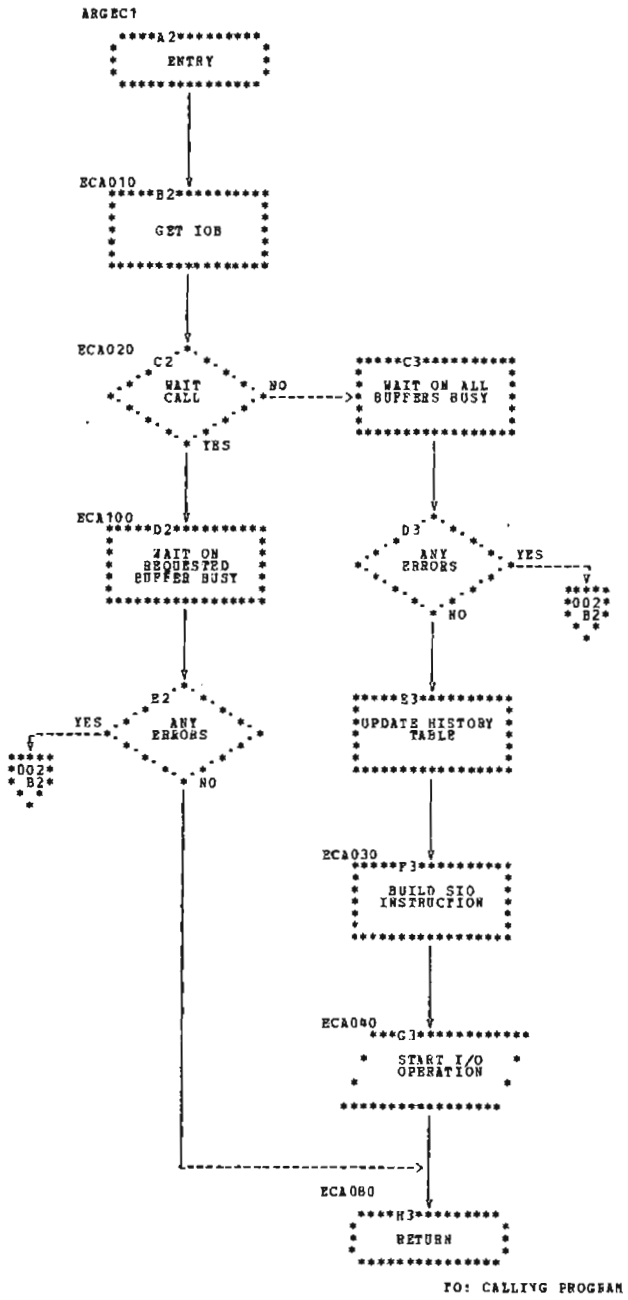


Chart IF. Read Primary (Object)–Wait and Execute Calls (Part 1 of 2)

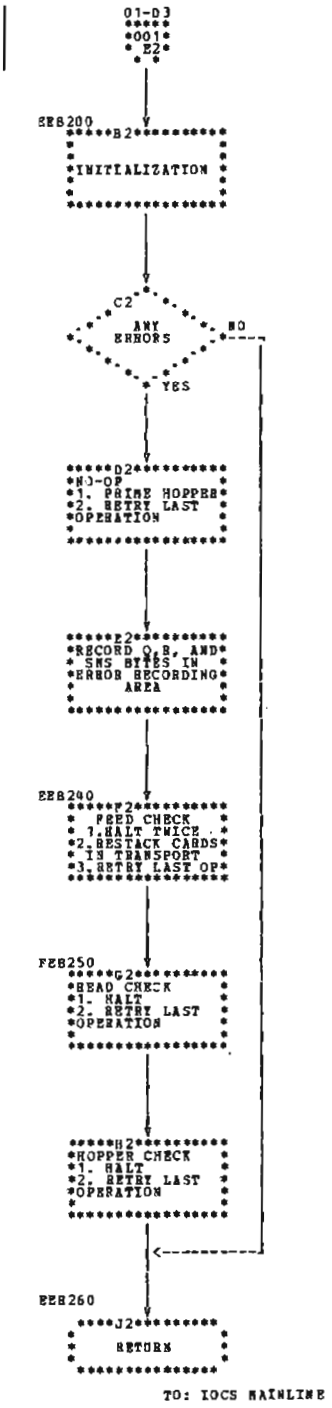


Chart IF. Read Primary (Object)–Error Recovery Procedures (Part 2 of 2)

Read Primary/Punch Secondary (Compiler)

Entry Points:

- ARGEK1 – Wait for cards to clear transport.
- ARGEK2 – Punch wait call.
- ARGEK3 – Punch execute call.
- ARGEK4 – Read execute and wait call.

Chart: IG

Functions:

- Waits for the I/O operation to complete.
- Waits for cards to clear transport.
- Reads cards from the primary hopper.
- Punches cards from the secondary hopper.
- Directs cards from the primary hopper to stacker 1.
- Directs cards from the secondary hopper to stacker 3.
- Detects and recovers from errors.

Input: None

Output: Punched output

Exit: Control is returned to the instruction immediately following the branch to this routine.

Routine Distinctions: All ERPs are provided except:

1. Print check.
2. Punch invalid.

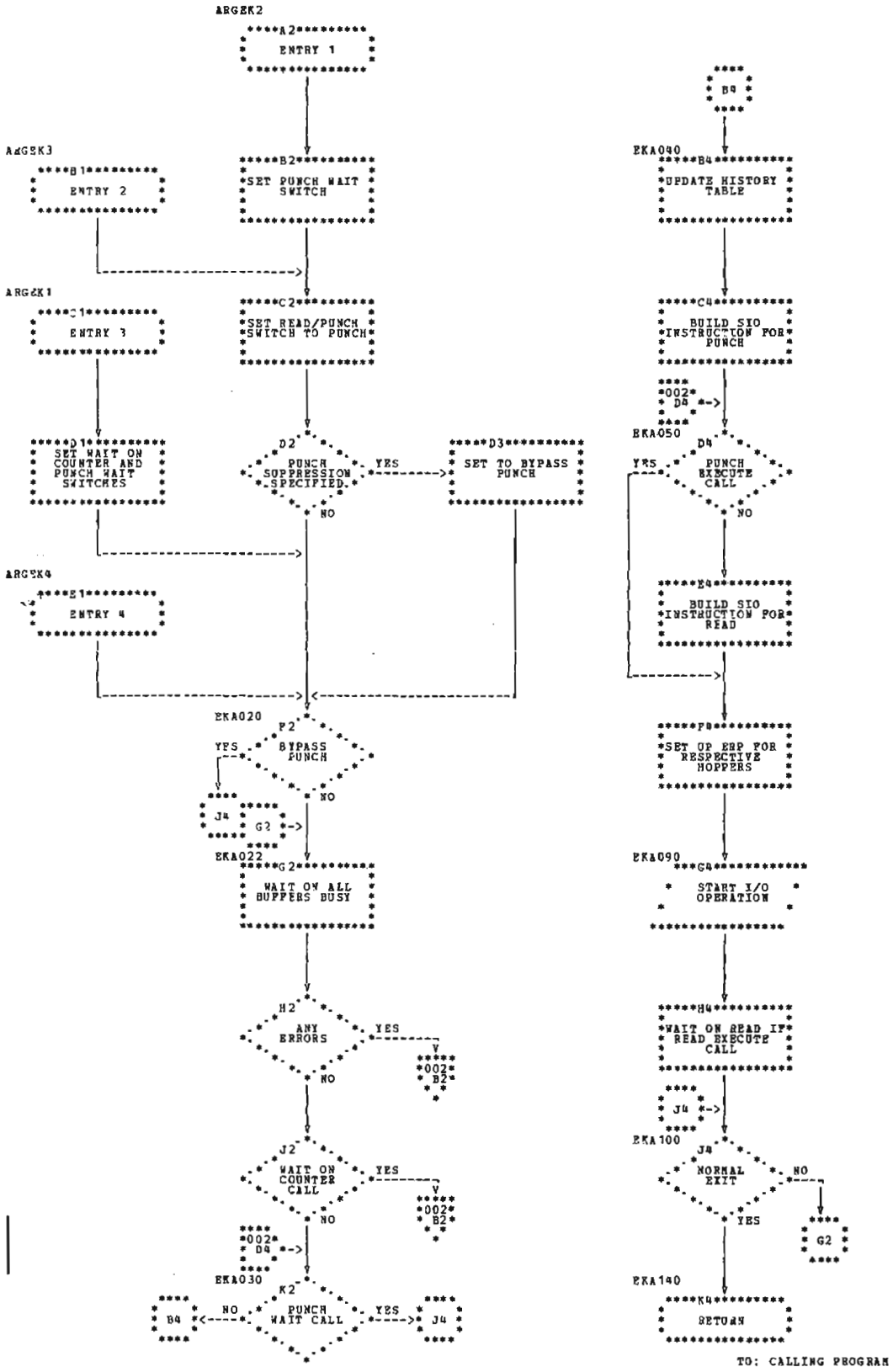


Chart IG. Read Primary/Punch Secondary (Compiler)–Wait and Execute Calls (Part 1 of 2)

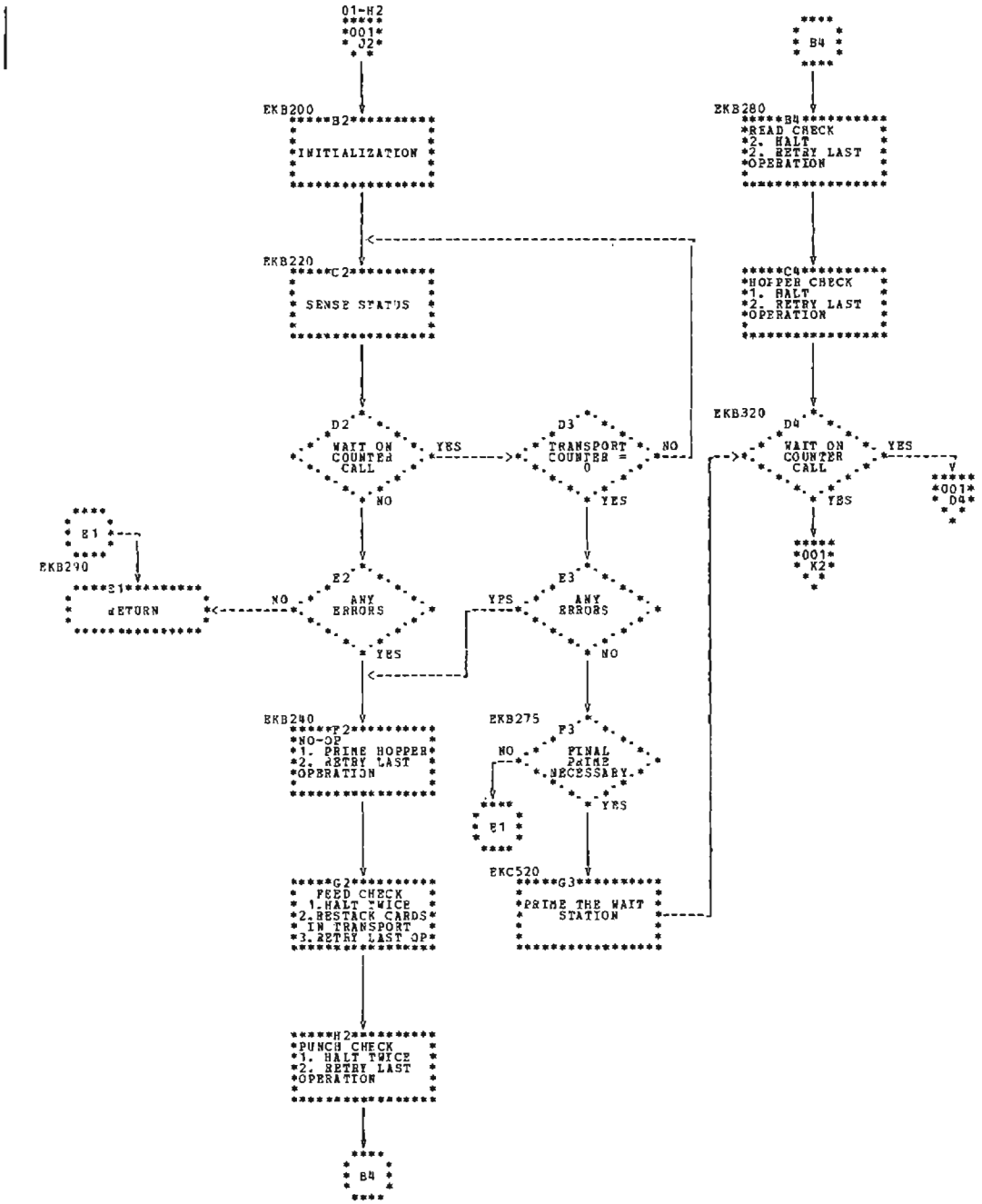


Chart IG. Read Primary/Punch Secondary (Compiler)--Error Recovery Procedures (Part 2 of 2)

Entry Points:

- ARGEB1 – Execute call
- ARGEB2 – Wait for cards to clear transport
- ARGEB3 – Wait call

Chart: IH

Functions:

- Waits for the I/O operation to complete.
- Waits for cards to clear transport.
- Reads cards from the secondary hopper.
- Directs cards to stacker 4.
- Detects and recovers from errors.

Input: None

Output: None

Exit: Control is returned to the instruction immediately following the branch to this routine.

Routine Distinctions: All ERPs are provided except:

1. Print check.
2. Punch invalid.
3. Punch check.

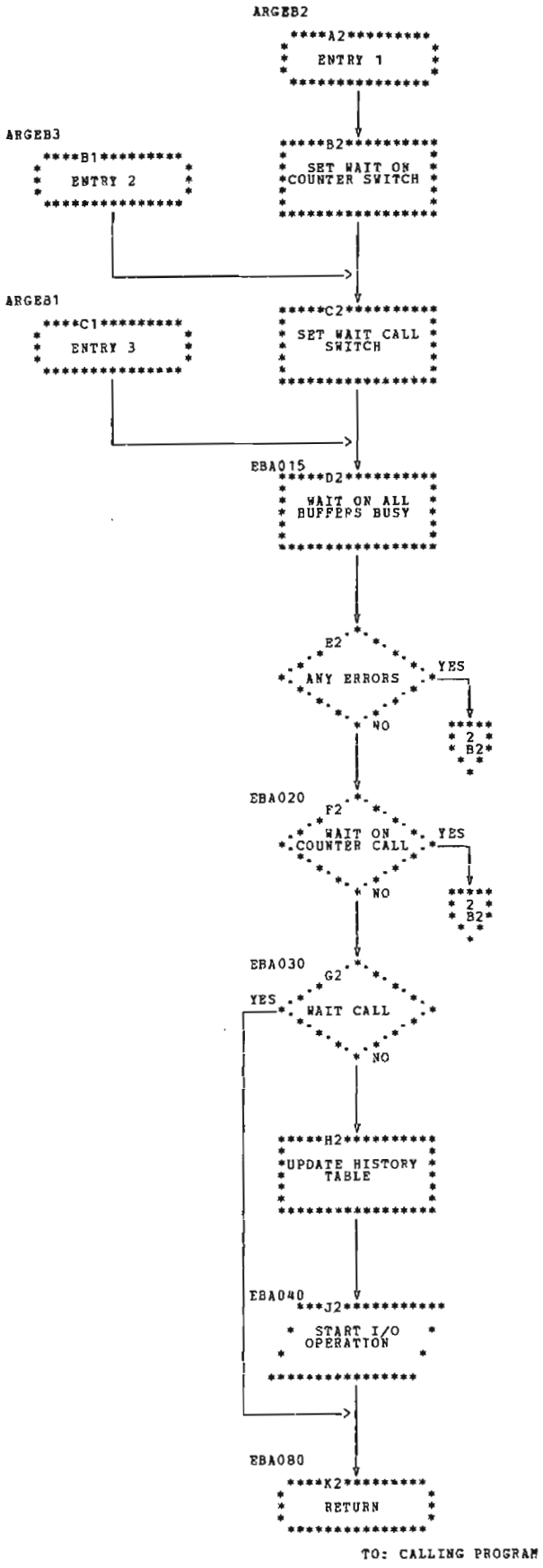


Chart IH. Read Secondary (Compiler)–Wait and Execute Calls (Part 1 of 2)

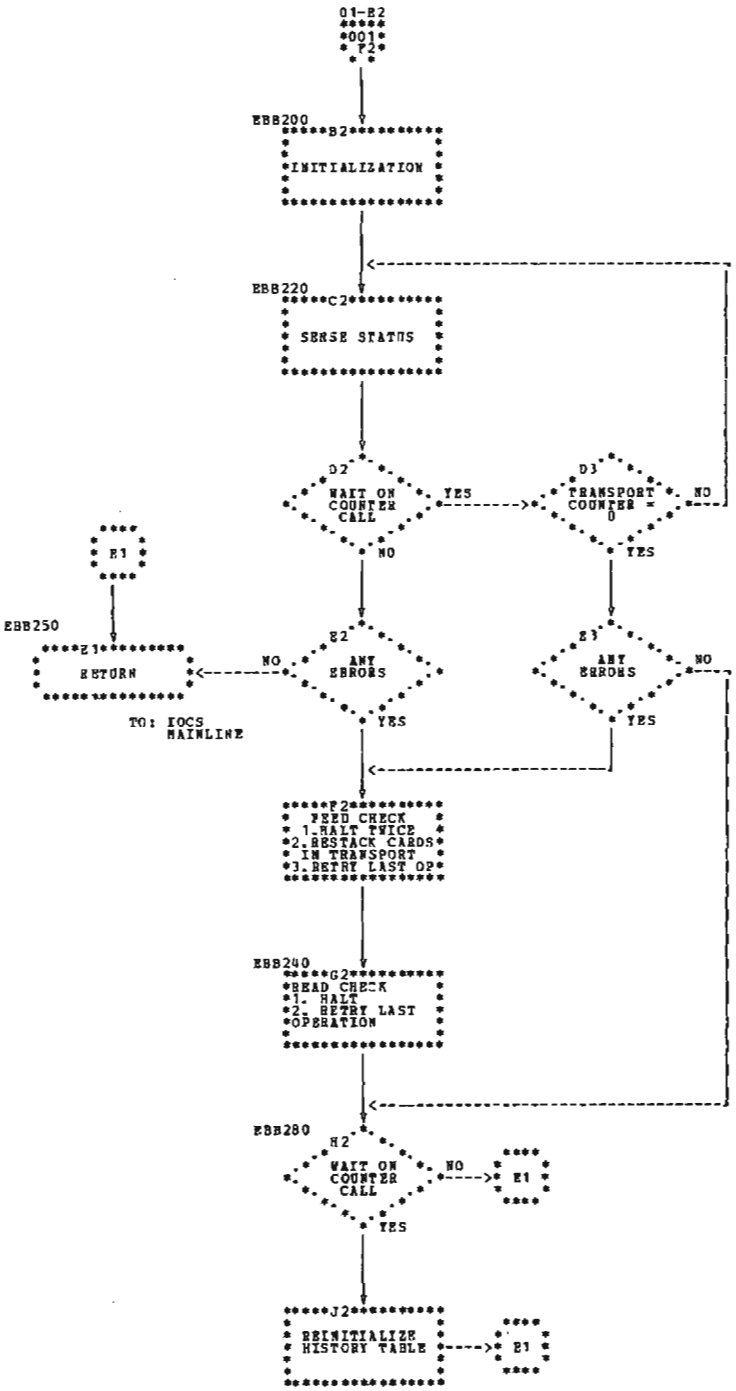


Chart IH. Read Secondary (Compiler)–Error Recovery Procedures (Part 2 of 2)

Read Only/Both Hoppers (Object)

Entry Point: ARGEE1

Chart: II

Functions:

- Waits for the I/O operation to complete.
- Reads cards from either the primary or secondary hopper.
- Provides stacker selection.
- Detects and recovers from errors.

Input:

- IOB
- DTF

Output: None

Exit: Control is returned to the instruction immediately following the 2-byte parameter pointing to the IOB.

Routine Distinctions:

All ERPs are provided except:

1. Print check.
2. Punch invalid.
3. Punch check.

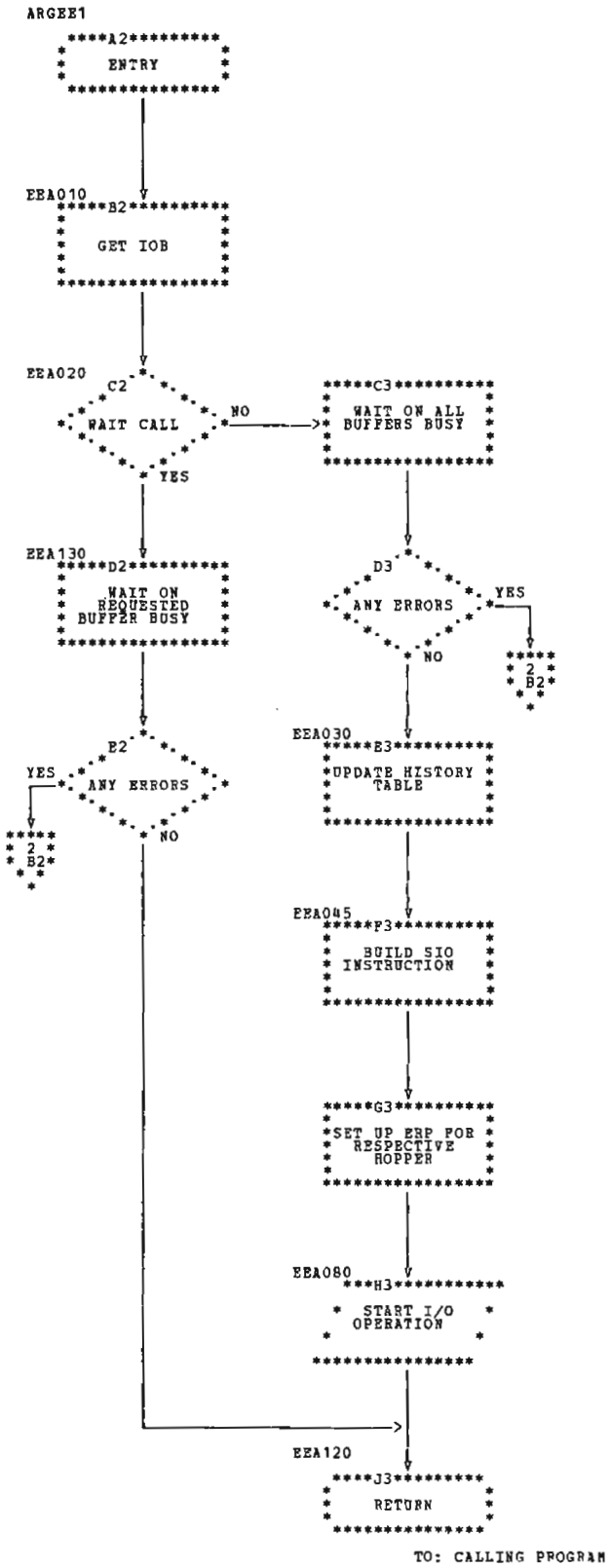


Chart II. Read Only/Both Hoppers (Object)--Wait and Execute Calls (Part 1 of 2)

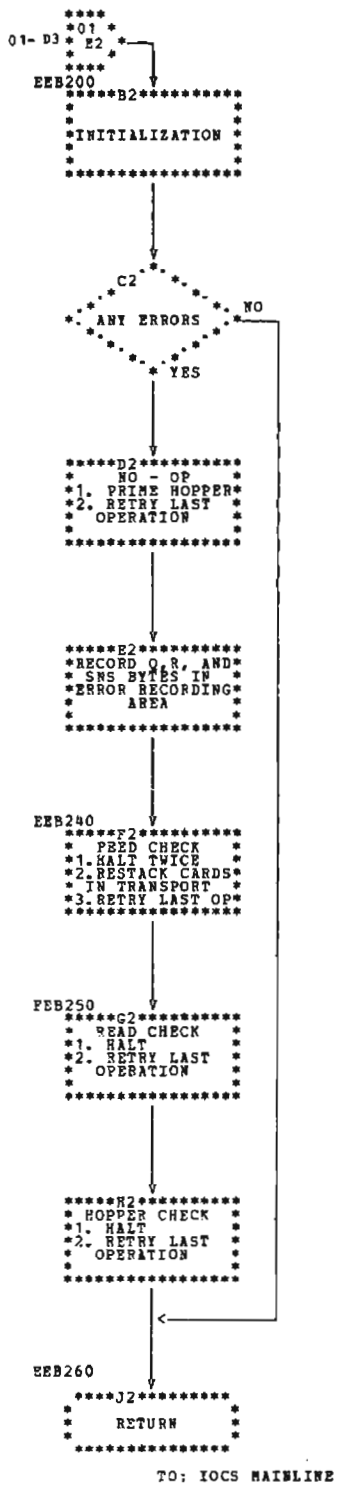


Chart II. Read Only/Both Hoppers (Object)—Error Recovery Procedures (Part 2 of 2)

Punch Only/Secondary Hopper (Compiler)

Entry Points:

- ARGEF1 – Execute call
- ARGEF2 – Wait for cards to clear transport
- ARGEF3 – Wait call

Chart: IJ

Functions:

- Waits for the I/O operation to complete.
- Waits for cards to clear transport.
- Punches cards from the secondary hopper.
- Directs cards to stacker 3.
- Detects and recovers from errors.

Input: None

Output: Punched output

Exit: Control is returned to the instruction immediately following the branch to this routine.

Routine Distinctions:

- All ERPs are provided except:
  1. Print check.
  2. Punch invalid.
  3. Read check.

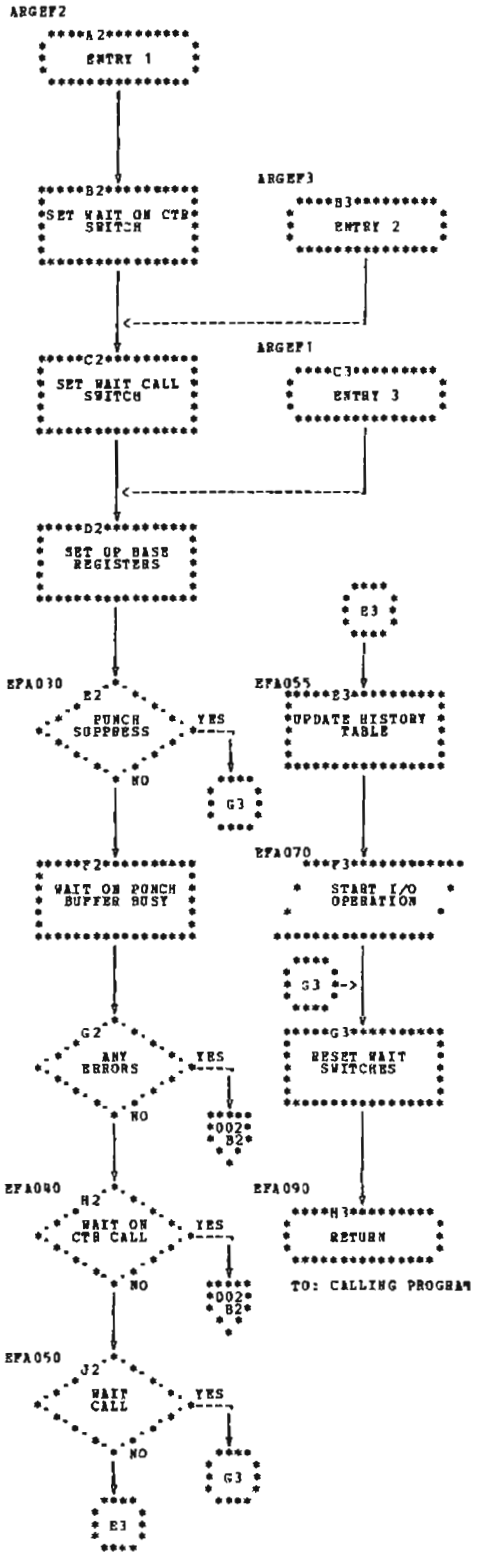


Chart IJ. Punch Secondary (Compiler)–Wait and Execute Calls (Part 1 of 2)

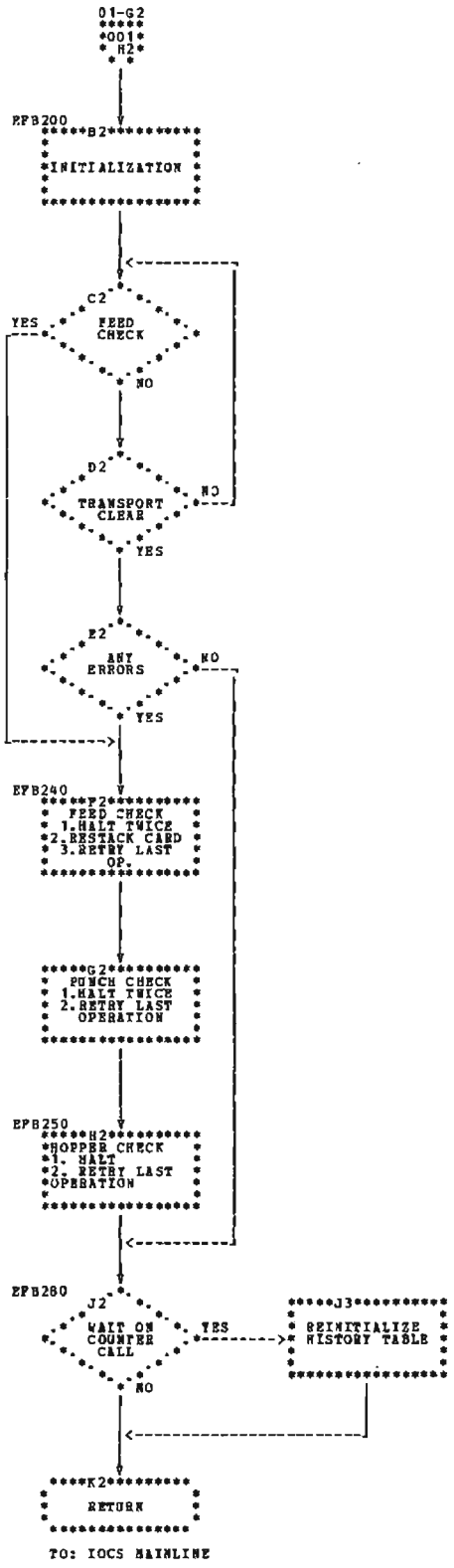


Chart IJ. Punch Secondary (Compiler)–Wait and Execute Calls (Part 2 of 2)

IOB (INPUT/OUTPUT BLOCK)

Information is passed between object programs and the MFCU IOCS routines with a 6-byte IOB. A 2-byte parameter immediately following the branch to the MFCU IOCS routines points to the low-order byte of the IOB. The contents of the IOB are shown in Figure 2-12.

DTF (DEFINE THE FILE)

The last two bytes of the IOB point to a 14-byte DTF. The DTF contains buffer addresses and error information. The contents of the DTF are:

Bytes	*Contents*
1-2	Address of the read buffer for the primary hopper.
3-4	Address of the read buffer for the secondary hopper.
5-6	Address of the punch buffer for the primary hopper.
7-8	Address of the punch buffer for the secondary hopper.
9-10	Address of the next print buffer to be used. This address is initialized to the first of two consecutive print buffers. This information is changed by IOCS at each print wait call.
11-12	Used for storage of Q and R bytes when an error occurs.
13-14	Used for storage of status bytes when an error occurs.

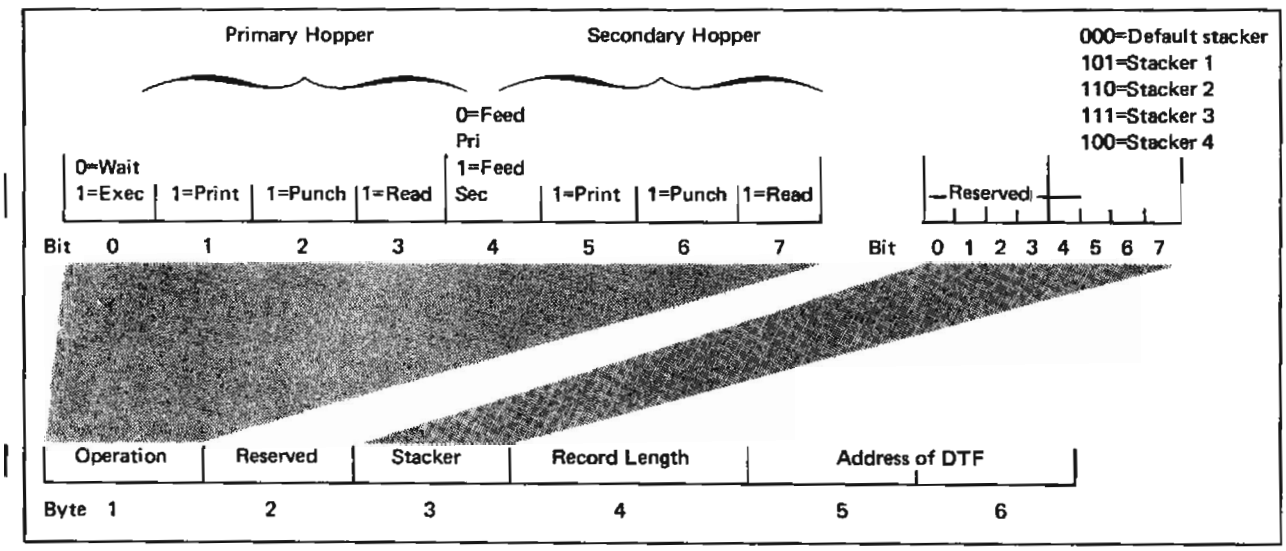


Figure 2-12. IOB Information for MFCU IOCS Routines

HISTORY TABLE

This History Table contains three bytes, one byte for each of the last three stackers selected. The build area for the R byte immediately precedes the History Table. When a stacker select occurs, the table is updated by shifting the R byte and the first two bytes in the table one byte to the right (Figure 2-13). All bits in the R byte which do not contain stacker information are set off. The History Table is referenced to determine the stacker number that should be displayed in the second halt (feed check, punch check, or punch invalid) or the stacker to be placed in the feed command (feed check or punch check).

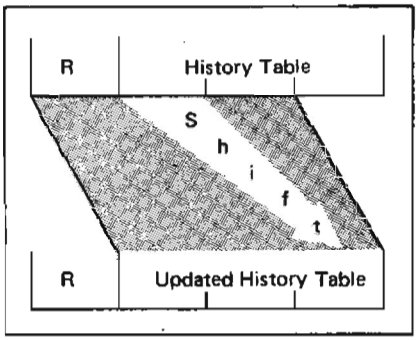


Figure 2-13. History Table

HALT TABLE

The 8-byte Halt Table contains the codes necessary to display the appropriate information for the second halt of a feed check, punch check, or punch invalid. The first four bytes contain the halt codes used to display the number of cards to be replaced in the hopper (0, 1, 2, or 3). The last four bytes contain the halt codes used to display the last selected stacker (4, 1, 2, or 3).

The display counter or R byte is used as a displacement to the Halt Table. The display counter will contain the following, depending on the number of cards to be replaced in the hopper:

Contents	*Displacement*	*Meaning*
0000	0	0 cards
0001	1	1 card
0010	2	2 cards
0011	3	3 cards

The stacker information bits in the R byte contain the following, depending on the selected stacker:

Contents	*Displacement*	*Meaning*
0100	4	Stacker 4
0101	5	Stacker 1
0110	6	Stacker 2
0111	7	Stacker 3

ERROR RECORDING AREA

This 42-byte area starts at location X'180'. This area, a history table of I/O device errors, 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 hammer echo checks. Each entry is converted from the value recorded by IOCS (the rightmost byte of the Line Printer Data Address Register) to the true print position. See the System Initialization program for more information.

Section 1. Introduction

The IBM System/3 Card System Input/Output Control System (IOCS) for the IBM 5203 Printer is described in this chapter. The IOCS routines perform I/O (input/output) operations requested by IBM System/3 Card System programs. Error recovery procedures and operator restart procedures are provided for each IOCS routine. Halt identifiers, which appear on the Message Display Unit, and operator restart procedures are described in the *IBM System/3 Card System Operator's Guide*, GC21-7513. Figure 2-14 shows which IOCS routines are used by the IBM System/3 Card System programs.

Card System Programs \ Printer IOCS Routines	Line Printer–Single Feed Carriage (Object)	Line Printer–Dual Feed Carriage (Object)	Line Printer-Single Feed Carriage (Compiler)
Program Maintenance	X		
Device Counter Logout			X
96 List	X		
Sort/Collate			X
RPG II Compiler			X
PRG II Compiler Object Program	X	X	

Note: These compiler phases contain the Line Printer-Single Feed Carriage (Compiler) routine in storage:

- RGAA
- RGAC
- RGAE
- RGAG
- RGAI
- RGAK
- RGAM
- RGAO
- RGAP-RGBZ
- RGCA
- RGCC
- RGCI
- RGCM
- RGCQ
- RGCS
- RGCU
- RGCW
- RGDE
- RGEA
- RGME

Figure 2-14. Printer IOCS Routines Used by Card System Programs

SYSTEM REQUIREMENTS

The IBM System/3 printer IOCS routines use the following system configuration:

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

Section 2. Method of Operation

The two major functions of the printer IOCS are to (1) identify and perform the requested IOCS function and (2) detect and recover from any printer I/O errors. Figure 2-15 shows the functional and data flow for object program IOCS requests. Figure 2-16 shows the functional and data flow for compiler IOCS requests.

PERFORM IOCS FUNCTION

Object Program

The calling program branches to the printer IOCS routine to request a printer function. An IOB (input/output block) passed by the calling program specifies either a wait or an execute call.

A wait call performs these functions:

- Waits for the print operation to complete by waiting until the buffer is not busy.
- Checks for errors and performs error recovery procedures if an error is detected.
- Checks for unprintable characters unless specified otherwise on the RPG II control card (the Program Maintenance and 96 List programs ignore unprintable characters). No error recovery is done. The halt code is PC.
- Clears the print buffer to blanks.

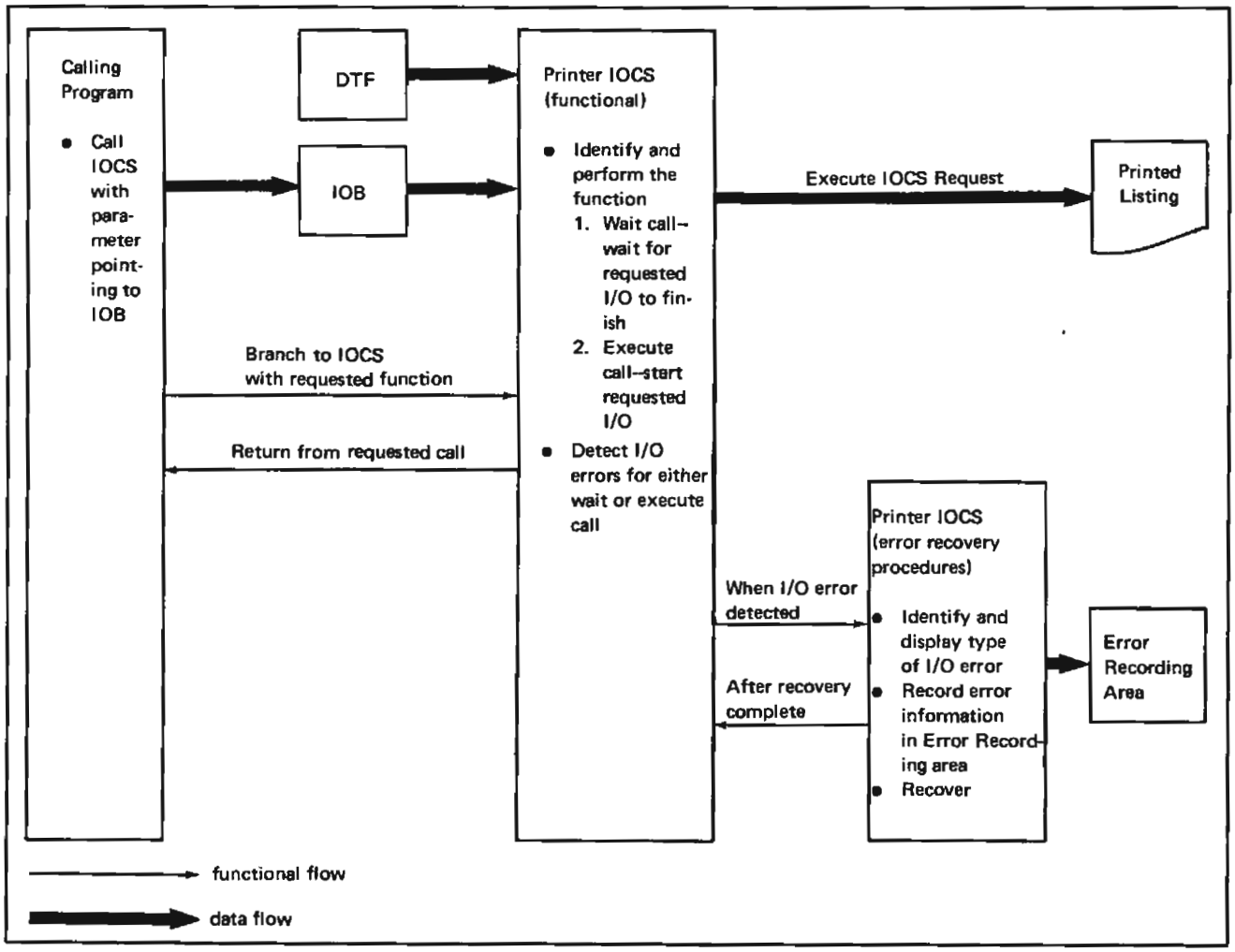


Figure 2-15. Functional and Data Flow for Object Program Printer IOCS Requests

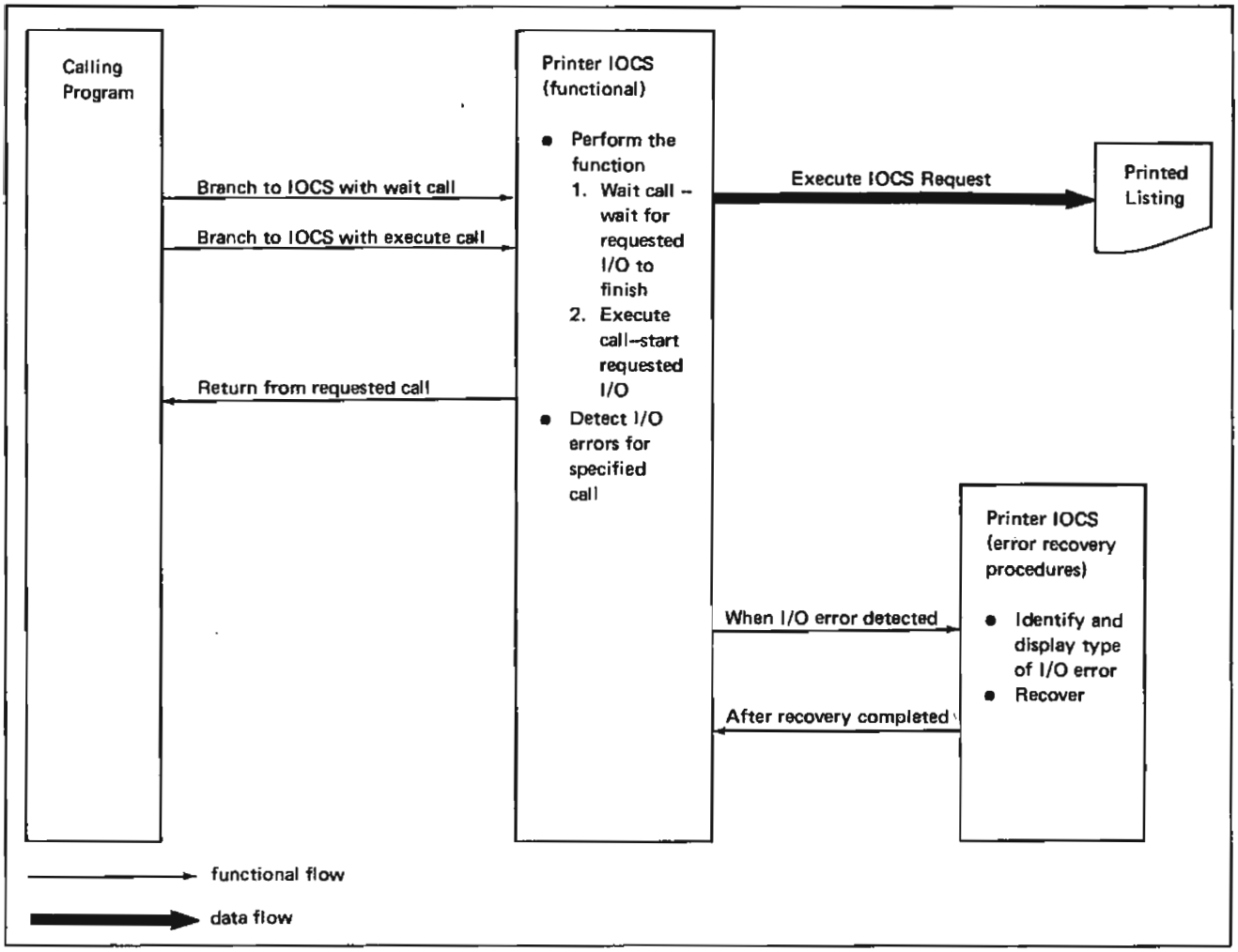


Figure 2-16. Functional and Data Flow for Compiler Printer IOCS Requests

An execute call performs these functions:

- Checks for errors and branches to the error recovery procedures if an error is detected.
- Attempts to start the requested I/O operation.

1. Execute call – Prints line and spaces one after.
2. Wait call – Waits for the print operation to complete.
3. Overflow call – Skips to line 6 of the next page.

A skip to the next page of the listing occurs if:

- Overflow call is requested.
- The line six less than the form length specified in the System Initialization Program is reached (see *IBM System/3 Card System System Control Programs Logic Manual*, SY21-0522).

Compiler

The compiler IOCS routine does not use an IOB or DTF (define the file). The entry point determines the function to be performed. These functions can be requested:

DETECT AND RECOVER FROM I/O ERRORS

Object Program

Both the wait call and execute call check for I/O errors. If an error is detected, the IOCS routine branches to the error recovery procedures (ERPs). The ERPs attempt to recover from an I/O error by re-trying as many of the previous operations as necessary to properly complete the operations previously requested and stack the paper.

Error information is recorded in the Error Recording area for all errors found. A halt code is displayed when the error is detected. For a detailed description of the ERPs, see Section 3, *Program Organization.* Figure 2-18 is a summary of the halt codes for the error types and the program action taken.

Printer IOCS recovers from six types of errors:

- Sync check – The halt code is P5. If an operation was attempted but could not be started, the operation is repeated. Otherwise, no program error recovery is done.
- Incrementer failure check – The halt code is P6. The last operation is repeated.
- Print check – The halt code is P8. A print check can be caused by a hammer on check or a hammer echo check. If this error is caused by a hammer echo check, the hammer echo check information in the Error Recording area is updated. The last operation is repeated for either type of print check.
- Thermal check – The halt code is P7. Error recovery for thermal check is the same as error recovery for sync check.
- Carriage check – The halt code is P1. The space/skip operation is repeated. If printing was specified in the I/O operation, it is not repeated because the printing has been completed.
- Forms jam – The halt code is P3. The skip operation is repeated. If printing was specified in the I/O operation, it is not repeated because the printing has been completed.

Compiler

The compiler IOCS ERP differs from object program IOCS ERPs in that it does not record error information in the Error Recording area.

Section 3. Program Organization

GENERAL DESCRIPTION OF PRINTER IOCS ROUTINES

The overview presented in this section deals with an object program IOCS routine. The compiler IOCS routine differs from the object program IOCS routines in these ways:

- Compiler IOCS routine does not use an IOB or DTF.
- Compiler IOCS routine has separate entry points. The entry points are:
  1. Execute call
  2. Wait call
  3. Overflow call
- Compiler IOCS routine checks for print suppression.
- Compiler IOCS routine does not record error information in the Error Recording area.

See *Printer IOCS Routines* in this section for flowcharts of specific routines.

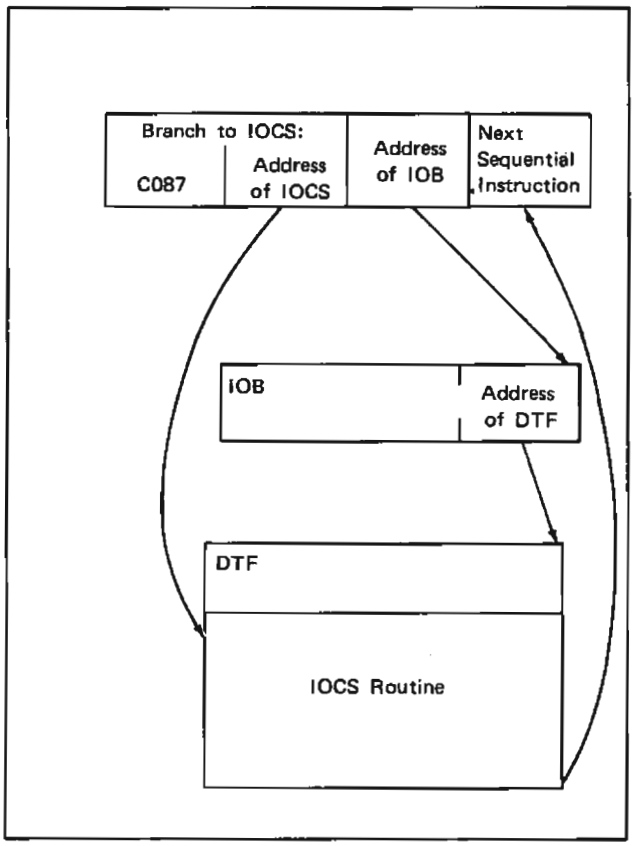


Figure 2-17. Linkage for Object Program Printer IOCS Routines

Functional Description

The calling program branches to the printer IOCS routines to request a printer function. Immediately following the branch is a parameter which points to the IOB. The IOB contains the type of call, the functions to be performed, and the space/skip values. The last two bytes of the IOB point to the 6-byte DTF. The DTF contains overflow line information and provides space for storage of error information (see Section 4, *Data Area Formats,* for the contents of the IOB and DTF). Figure 2-17 shows the linkage of these areas.

Before checking whether a wait call or execute call was requested, IOCS performs these functions:

- Initializes registers.
- Waits for print buffer to be not busy.
- Checks for errors (incrementer failure, print checks, sync check, and thermal check) and performs error recovery procedures if an error is detected.

Wait Call

If a wait call is requested, IOCS performs the following steps to complete the functions:

- Checks for unprintable characters unless specified otherwise on the RPG II control card. The halt code is displayed (PC). The program continues processing. Since this check is done during the wait call, a wait call must follow every print command if the check is desired.
- Clears the print buffer to blanks.

IOCS then returns to the calling program. Chart IK shows an overview of the wait call. For a more detailed flowchart, see *Detailed Description of the Printer IOCS Routines* in this section.

Execute Call

If an execute call is requested, IOCS performs the following steps to complete the functions:

- Waits for all previous I/O operations on the printer to complete.
- Checks for all device errors and performs error recovery procedures if an error is detected.
- Sets overflow indicators for space or skip operations as specified by the RPG II source program. The overflow indicators in the utility programs are set for automatic overflow.
- Builds the SIO instruction by moving in the Q byte and the R byte.
- Initiates the I/O operations (SIO).

When the I/O operation has been completed for a wait call or accepted for an execute call, IOCS returns to the calling program. Chart IK shows an overview of the execute call. For a more detailed flowchart, see *Detailed Description of the Printer IOCS Routines* in this section.

Error Recovery Procedures Description

The wait call or execute call performs the error recovery procedures when it detects a device error. The status bytes are sensed into the DTF and loaded into the ARR (address recall register) from the DTF. Figure 2-18 shows the status sense information for each error.

A check is then made for I/O errors. If none have occurred, the error is a simple device-not-ready; the IOCS ERP returns to the IOCS mainline routine.

The Q, R, and SNS bytes in the DTF are moved into the Error Recording area. See Section 4, *Data Area Formats*, for a description of the Error Recording area.

Halt Code	Condition	Priority	Status		I/O Attention Light	Program Action
			Byte	Bit		
PC	Unprintable character	—	1	6	Off	Continue processing.
None	Simple not ready	—	—	—	On	Loop on the SIO until it is accepted.
P5	Sync check (see Note 3)	1,2	1	0,1	Off	Continue processing.
P6	Incrementer failure check	3	2	3	Off	Re-execute SIO.
P8	Print check (see Note 3)	4,5	2	5,6	Off	Re-execute SIO.
P7	Thermal check	6	1	2	Off	Continue processing.
P1	Carriage check (see Note 3)	7,8	2	0,1	Off	Re-execute carriage control portion of SIO (set off print bit).
P3	Forms jam	9	2	2	Off	Re-execute SIO. Skip to proper line.
None	No-op	—	2	7	Off	————

Note 1: Errors are processed in the order they appear on this chart except for no-op. No-op is checked following a sync check and a thermal check.

Note 2: Byte 1 refers to the second byte (high storage address) of sense data.

Note 3: Some device errors have two types of checks:

- Sync check
  1. Chain sync
  2. Incrementer sync
- Print check
  1. Hammer echo
  2. Hammer on
- Carriage check
  1. Carriage sync
  2. Carriage space

Figure 2-18. Error Recovery Information for Printer IOCS Routines

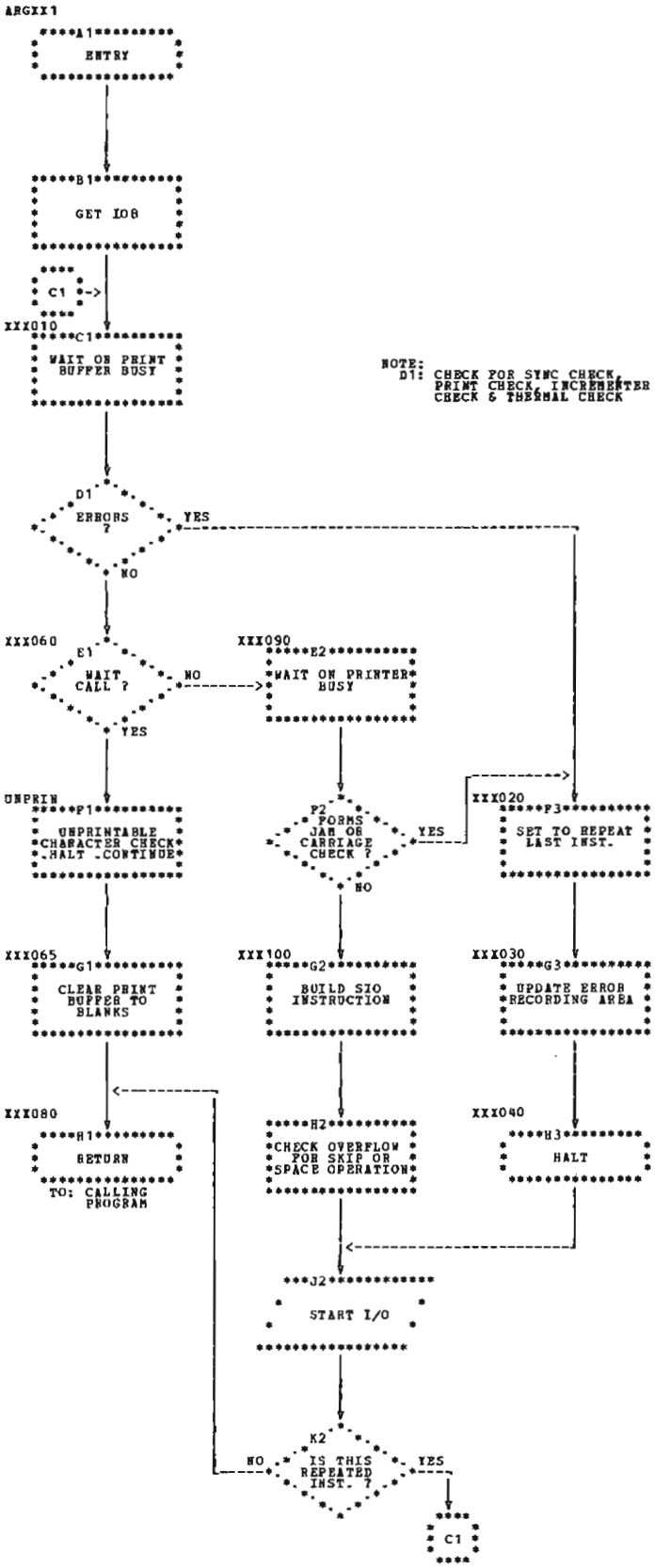


Chart 1K. Overview—Printer IOCS

Six types of errors are checked for in this order:

1. Sync check
2. Incrementer failure check
3. Print check
4. Thermal check
5. Carriage check
6. Forms jam

If there are no errors, a simple device-not-ready has occurred; this does not require program error recovery. Device-not-ready can be caused by:

1. No forms.
2. Covers open.
3. Printer STOP key pressed.

Chart IK shows an overview of the ERPs. For a more detailed flowchart, see *Detailed Description of the Printer IOCS Routines* in this section.

Sync Check and Thermal Check

Error recovery is the same for these two errors. The correct halt code is displayed (P5 or P7). If the SIO instruction was initiated after the error occurred, as indicated by the no-op bit being on, the program returns to the IOCS mainline to initiate the SIO instruction again. If the error occurred before the SIO instruction was built, the program returns to the IOCS mainline to continue processing. Figure 2-19 shows a schematic diagram of the sync check and thermal check ERP.

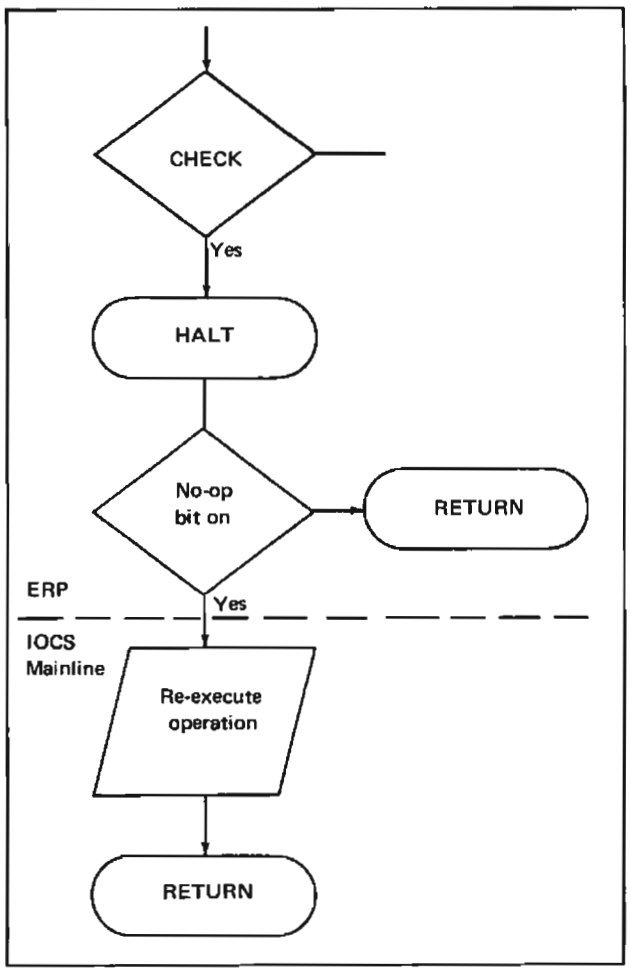


Figure 2-19. Schematic Diagram of Sync Check or Thermal Check ERP

Incrementer Failure Check

The halt code for an incrementer failure check is displayed (P6). The last operation is repeated. If no errors occur, the program returns to the IOCS mainline to continue processing. Figure 2-20 shows a schematic diagram of the incrementer failure check ERP.

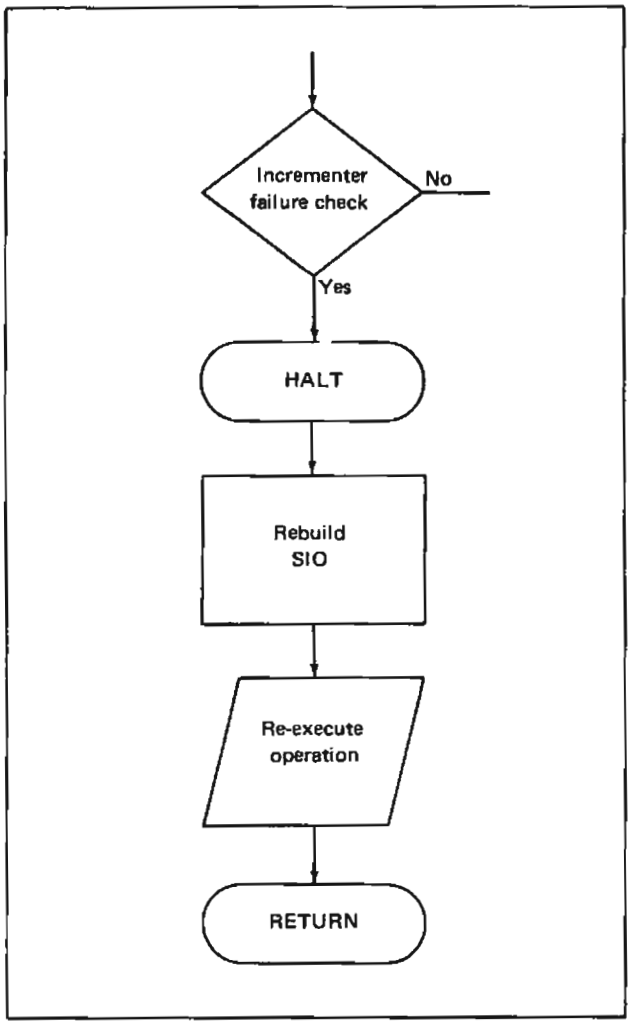


Figure 2-20. Schematic Diagram of Incrementer Failure Check ERP

Print Check

The program determines if this error was caused by a hammer echo check. If it is, the latest position in error is moved into the hammer echo check portion of the Error Recording area. The halt code for a print check is displayed (P8). The last operation is repeated. Figure 2-21 shows a schematic diagram of the print check ERP.

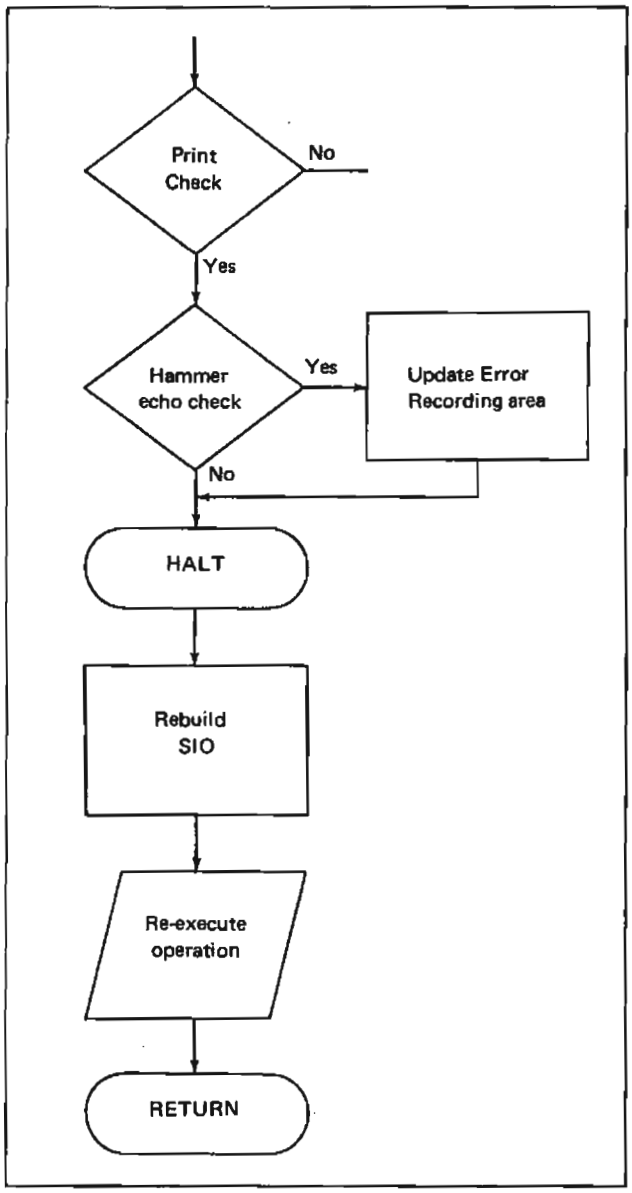


Figure 2-21. Schematic Diagram of Print Check ERP

Carriage Check

The print bit is set off so printing will not be repeated. The halt code for a carriage check is displayed (P1). The last operation, except printing, is repeated. Figure 2-22 shows a schematic diagram of the carriage check ERP.

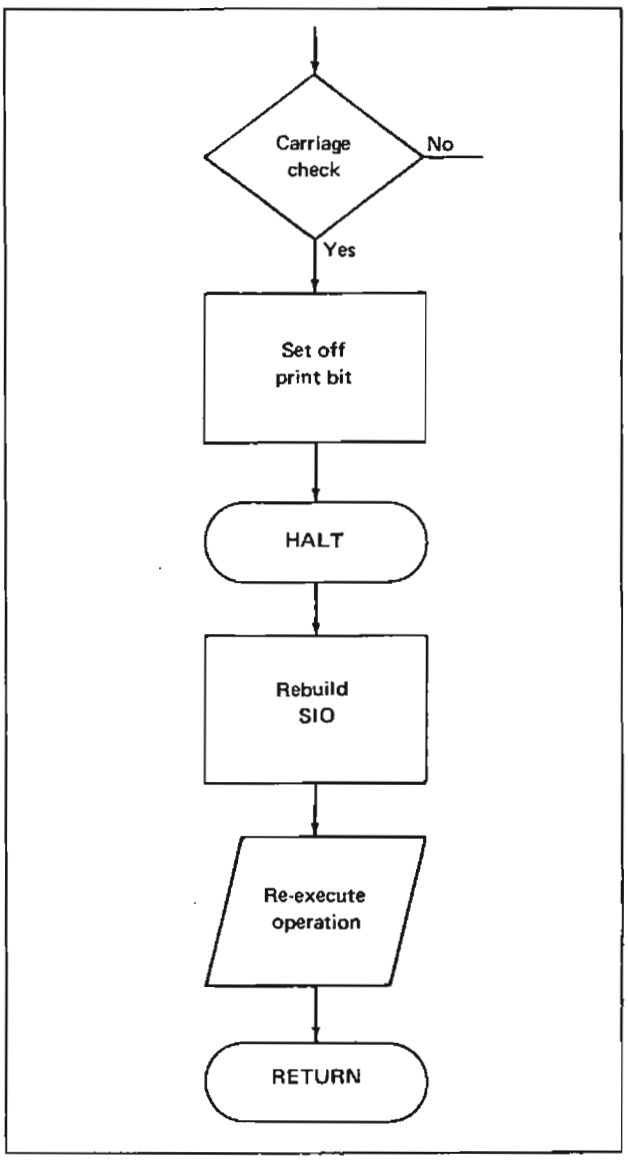


Figure 2-22. Schematic Diagram of Carriage Check ERP

Forms Jam

The print bit is set off so printing will not be repeated. Control and skip information are placed in the SIO instruction. The halt code for a forms jam is displayed (P3). The last operation, except printing, is repeated. Figure 2-23 shows a schematic diagram of the forms jam ERP.

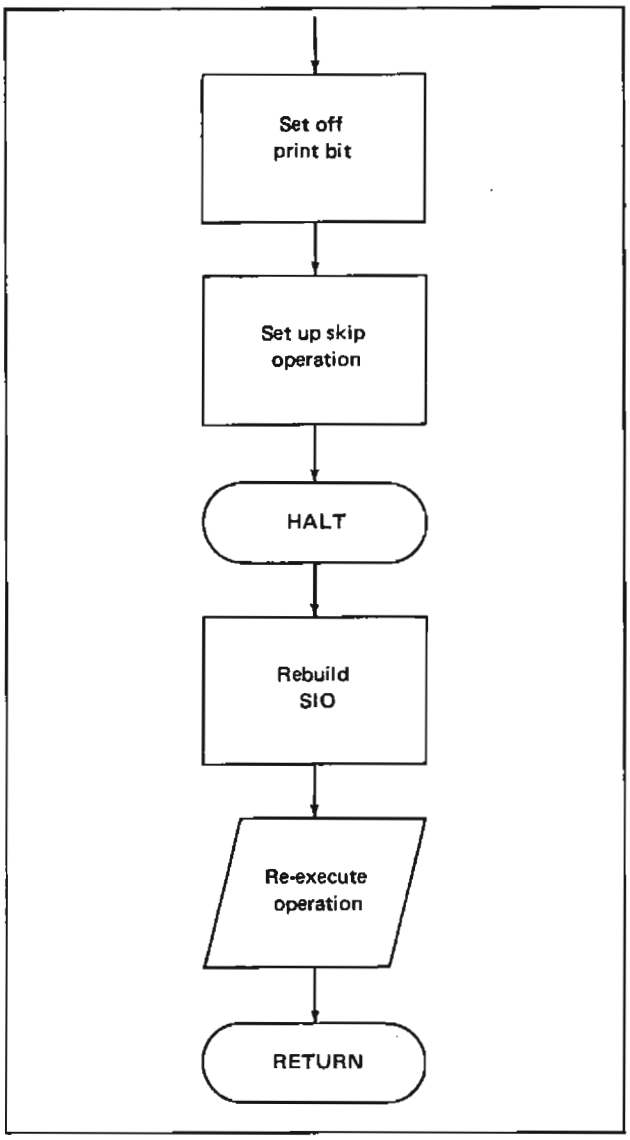
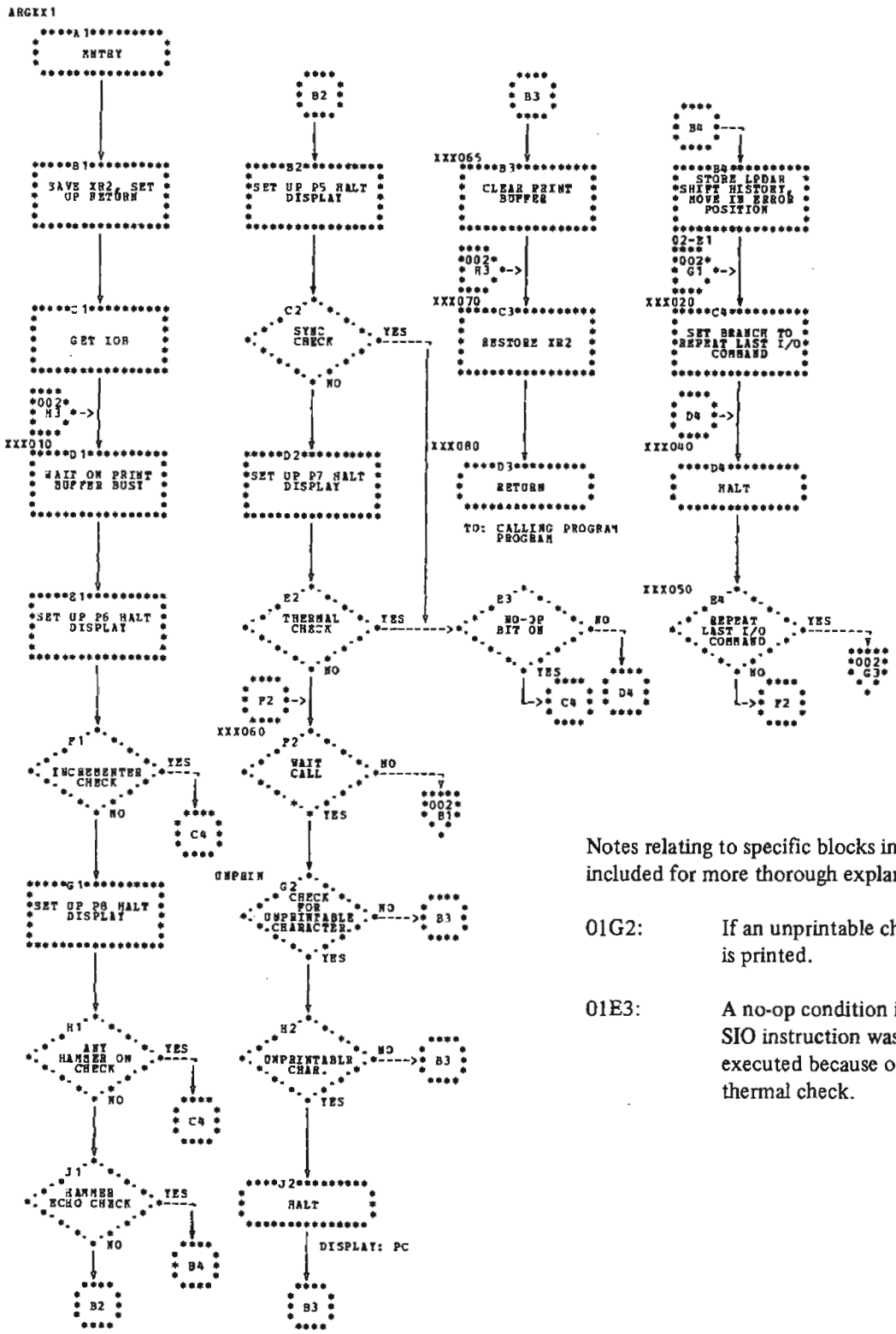


Figure 2-23. Schematic Diagram of Forms Jam ERP

DETAILED DESCRIPTION OF THE PRINTER IOCS ROUTINES

This section contains detailed flowcharts of the overview flowchart presented previously (Chart IK).

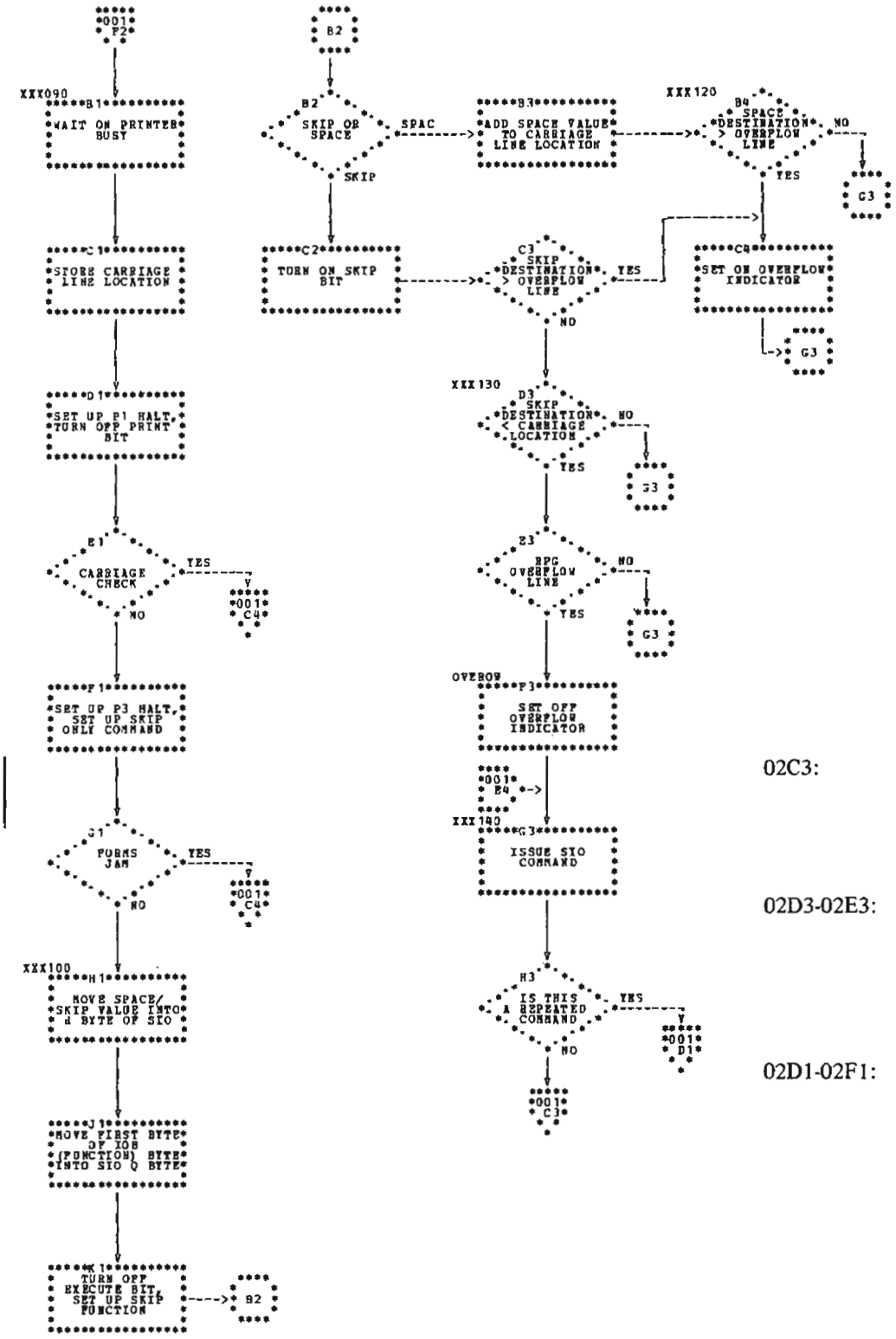


Notes relating to specific blocks in the flowcharts are included for more thorough explanations.

01G2: If an unprintable character is found, a blank is printed.

01E3: A no-op condition indicates that the last SIO instruction was accepted but not executed because of a sync check or a thermal check.

Chart IL. Detailed Description of Line Printer IOCS (Part 1 of 2)



02C3: If the form is positioned beyond the overflow line after a skip or space operation, overflow is indicated.

02D3-02E3: If the skip is to a new page for an RPG II overflow line, the overflow indicator is set off; otherwise, the indicator is unchanged.

02D1-02F1: The print bit is set off because the operation has already been completed and the print buffer has been cleared. The program skips to the proper line, but lost print lines cannot be recovered.

Chart IL. Detailed Description of Line Printer IOCS (Part 2 of 2)

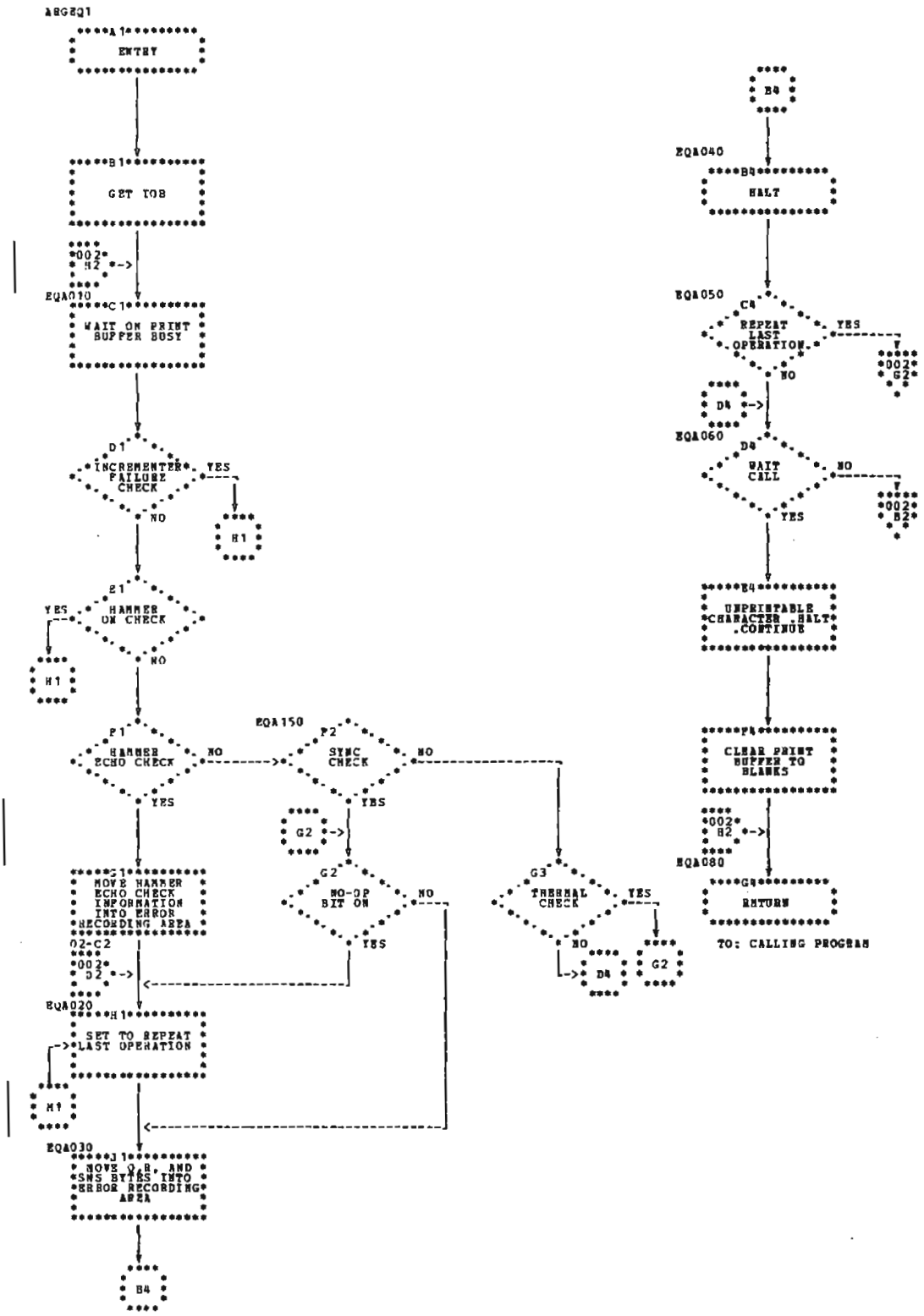


Chart IM. Line Printer–Single Feed Carriage (Object)–Execute Call and Error Recovery Procedures (Part 1 of 2)

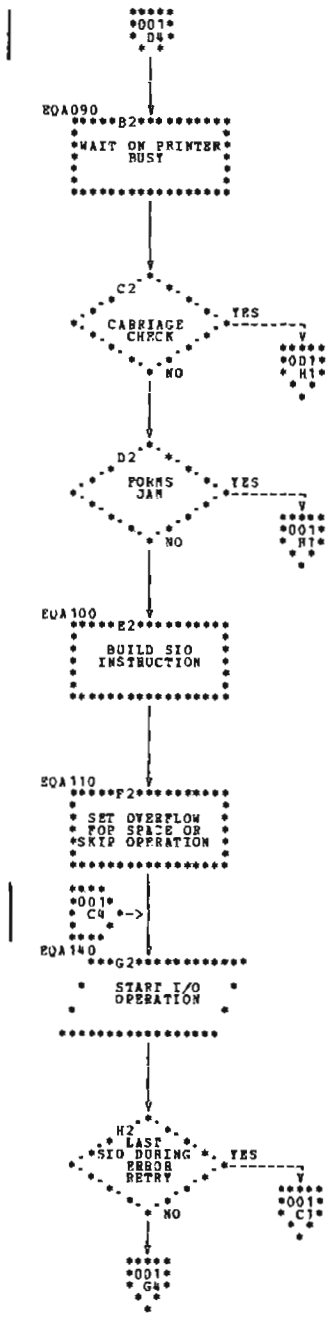


Chart IM. Line Printer–Single Feed Carriage (Object)–Execute Call and Error Recovery Procedures (Part 2 of 2)

PRINTER IOCS ROUTINES

This section gives a detailed description of each printer IOCS routine. The functions of each routine are given as well as its entry points, input required to perform the functions, resulting output, and exits. Distinctions from the general IOCS routine previously described are indicated. The flowcharts for a routine are on a level similar to Chart IK in *General Description of Printer IOCS Routines.* For a more detailed flowchart and discussion, see Chart IL.

Line Printer – Single Feed Carriage (Object)

Entry Point: ARGEQI

Chart: IM

Functions:

- Waits for the print buffer to be not busy.
- Initiates all valid printer operations for single feed carriage control.
- Detects and recovers from errors.

Input:

- IOB
- DTF

Output: Printed output

Exit: Control is returned to the instruction immediately following the 2-byte parameter pointing to the IOB.

Routine Distinctions: The order of error checking is changed, but the program recovery procedures are the same as described in Chart IL.

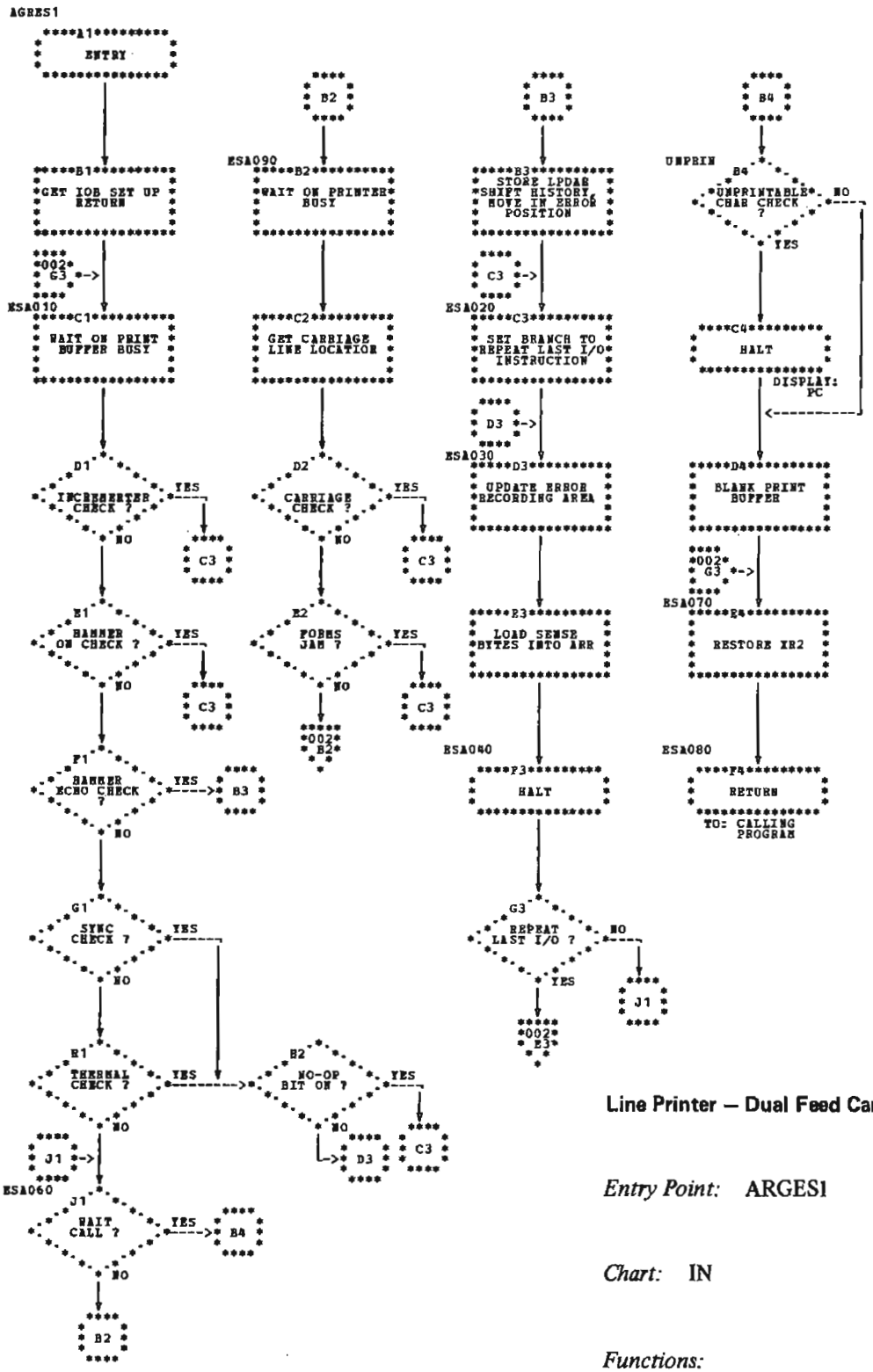


Chart IN. Line Printer–Dual Feed Carriage (Object) (Part 1 of 2)

Line Printer – Dual Feed Carriage (Object)

Entry Point: ARGES1

Chart: IN

Functions:

- Waits for the print buffer to be not busy.
- Initiates all valid printer operations for dual feed carriage control.
- Detects and recovers from errors.

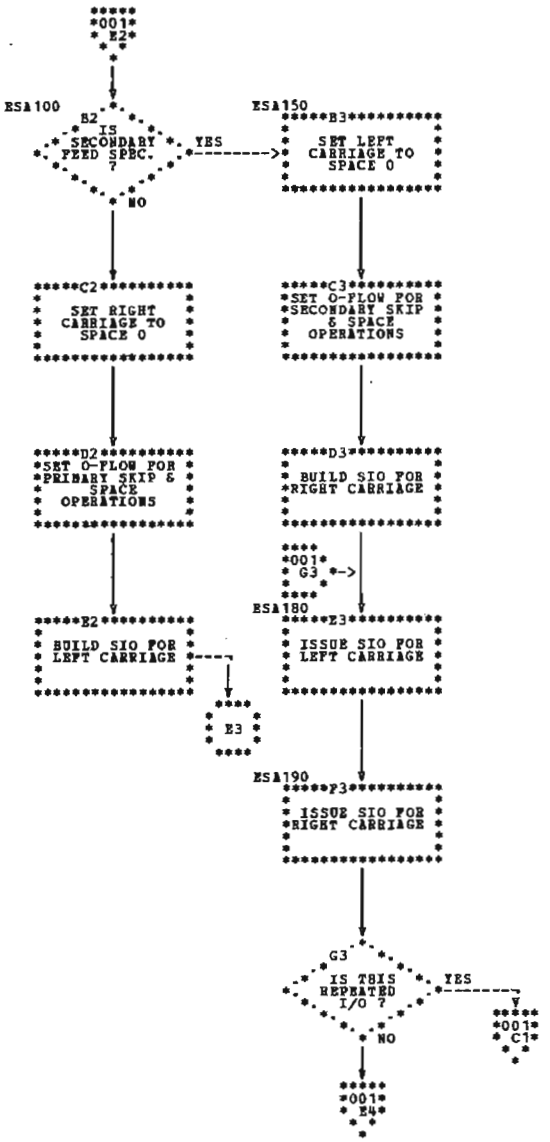


Chart IN. Line Printer–Dual Feed Carriage (Object) (Part 2 of 2)

Input:

- IOB
- DTF

Output: Printed output

Exit: Control is returned to the instruction immediately following the 2-byte parameter pointing to the IOB.

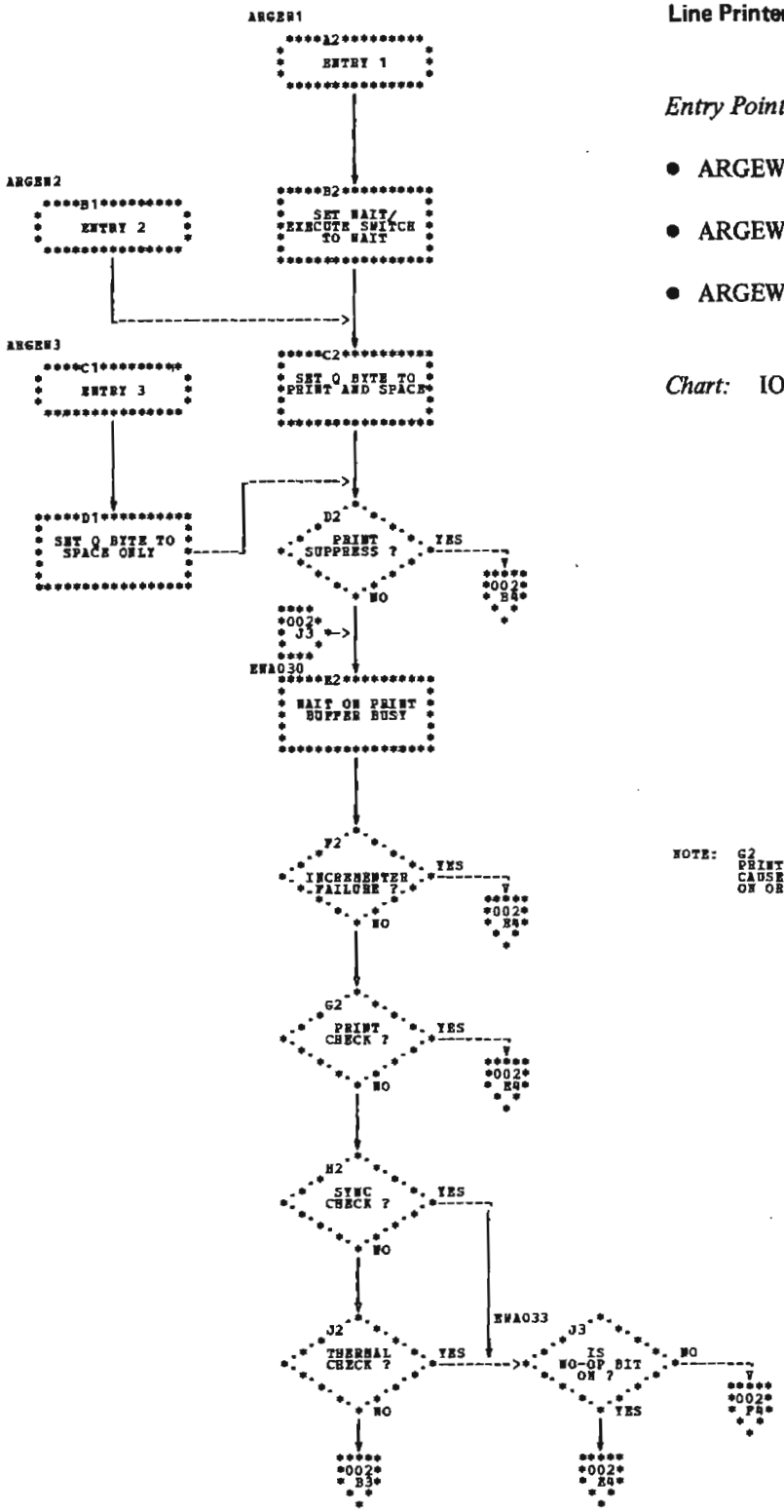
Routine Distinctions: Recovers from errors on either carriage.

Line Printer – Single Feed Carriage (Compiler)

Entry Points:

- ARGEW1 – Wait call
- ARGEW2 – Execute call
- ARGEW3 – Overflow call

Chart: IO



NOTE: G2
PRINT CHECK CAN BE CAUSED BY EITHER HAMMER ON OR HAMMER ECHO CHECK.

Chart IO. Line Printer–Single Feed Carriage (Compiler) (Part 1 of 2)

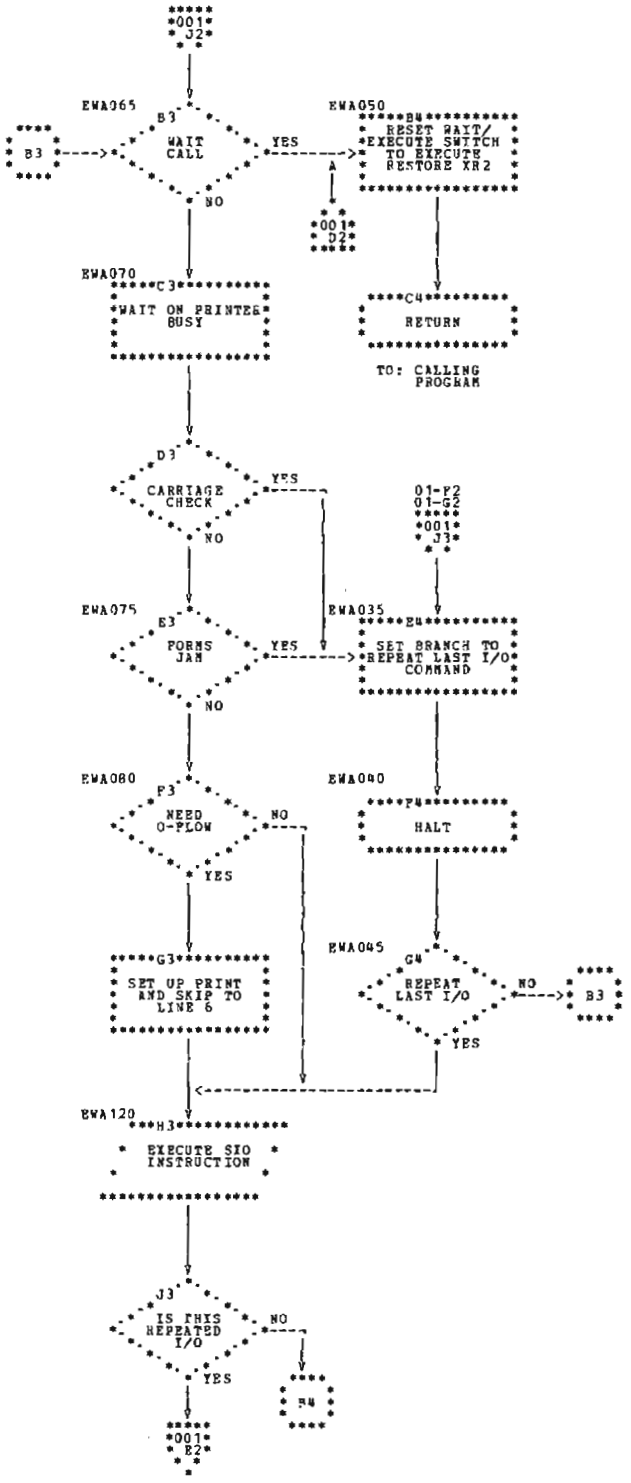


Chart IO. Line Printer–Single Feed Carriage (Compiler) (Part 2 of 2)

Functions:

- Waits for the print buffer to be not busy.
- Initiates all valid printer operations for single feed carriage control.
- Provides a skip to the next page of the listing on an overflow call or if the line six less than the form length specified in the System Initialization Program is reached (see *IBM System/3 Card System System Control Programs Logic Manual*, SY21-0522).
- Detects and recovers from errors.

Input: None

Output: Printed output.

Exit: Control is returned to the instruction immediately following the branch to this routine.

Routine Distinctions:

- Halt for unprintable characters is not provided.

IOB (INPUT/OUTPUT BLOCK)

Information is passed between object programs and the printer IOCS routine with a 6-byte IOB. A 2-byte parameter immediately following the branch to the printer IOCS routines points to the low-order of the IOB. The contents of the IOB are shown in Figure 2-24.

DTF (DEFINE THE FILE)

The last two bytes of the IOB point to a 6-byte DTF. The contents of the DTF are:

Byte	*Contents*
1	Left carriage overflow line value.
2	Right carriage overflow line value (dual carriage IOCS only).
3-4	Used for storage of Q and R bytes.
5-6	Used for storage of status bytes when an error occurs.

ERROR RECORDING AREA

This 42-byte area starts at location X'180'. This area, a history table of I/O device errors, 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) to the true print position. See the System Initialization program for more information.

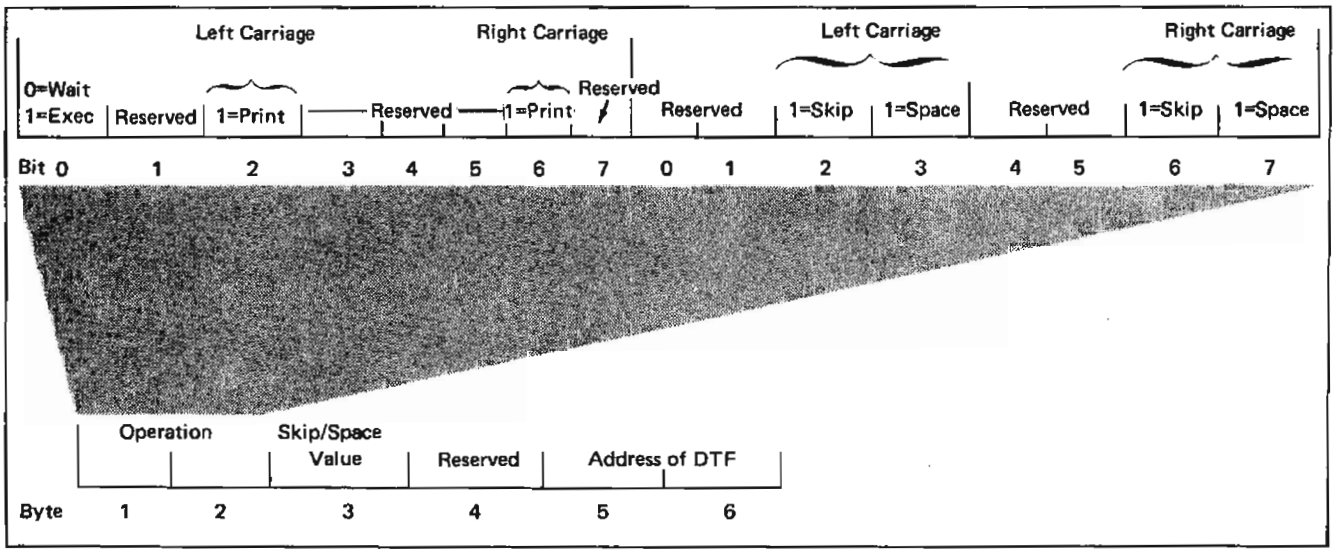


Figure 2-24. IOB Information for Printer IOCS Routines

Section 1. Introduction

The IBM System/3 Card System Input/Output Control System (IOCS) for the IBM 1442 Card Read-Punch is described in this chapter. The IOCS routines perform I/O (input/output) operations requested by IBM System/3 Card System programs. Error recovery procedures and operator restart procedures are provided for each IOCS routine. Halt identifiers, which appear on the Message Display Unit, and operator restart procedures are described in the *IBM System/3 80-96 Conversion Program and RPG II Support for the IBM 1442 Card Read-Punch*, SC21-7518. Figure 2-25 shows which IOCS routines are used by the IBM System/3 Card System programs.

SYSTEM REQUIREMENTS

The IBM System/3 IOCS routines for 1442 use the following system configurations:

- IBM 5410 Processing Unit
- IBM 5424 Multi-Function Card Unit (MFCU)
- IBM 1442 Model 6 Card Read-Punch

1442 IOCS Routines / Card System Programs	Read/Punch-No Feed (Object)	Read Column Binary (Object)	Punch-Feed (Object)	Read Only (Object)	Read Only (Compiler)
80-96 Conversion		X			
RPG II Compiler					X
RPG II Compiler Object Program	X		X	X	

Note: These compiler phases contain the Read Only (Compiler) routine in storage:

- RGAA
- RGAC
- RGAE
- RGAG
- RGAI
- RGAK
- RGCA
- RGCB
- RGCI
- RGCJ

Figure 2-25. 1442 IOCS Routines Used by Card System Programs

Section 2. Method of Operation

The two major functions of the 1442 IOCS are to (1) identify and perform the requested IOCS function and (2) detect and recover from any 1442 I/O errors. Figure 2-26 shows the functional and data flow for object program IOCS requests. Figure 2-27 shows the functional and data flow for compiler IOCS requests.

PERFORM IOCS FUNCTION

Object Program

The calling program branches to the 1442 IOCS routine to request a 1442 function. An IOB (input/output block) passed by the calling program specifies either a wait or an execute call.

A wait call performs these functions:

- Waits for the previous I/O operation on the 1442 to complete.
- Checks for errors and branches to the error recovery procedures if an error is detected.
- Clears the punch buffer to blanks if a punch call is requested.

An execute call performs these functions:

- Checks for errors and branches to the error recovery procedures if an error is detected.
- Attempts to start the requested I/O operation.

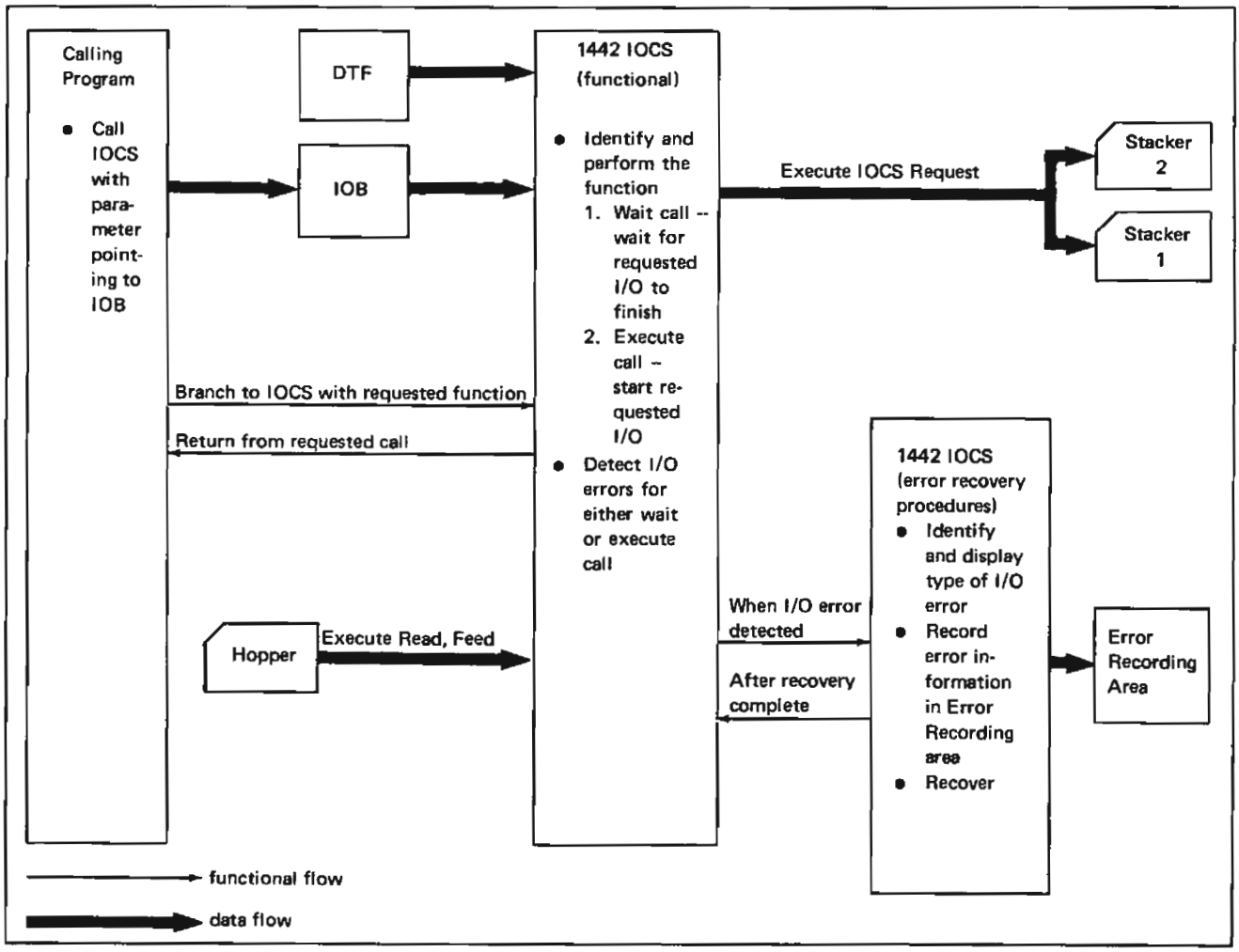


Figure 2-26. Functional and Data Flow for Object Program 1442 IOCS Requests

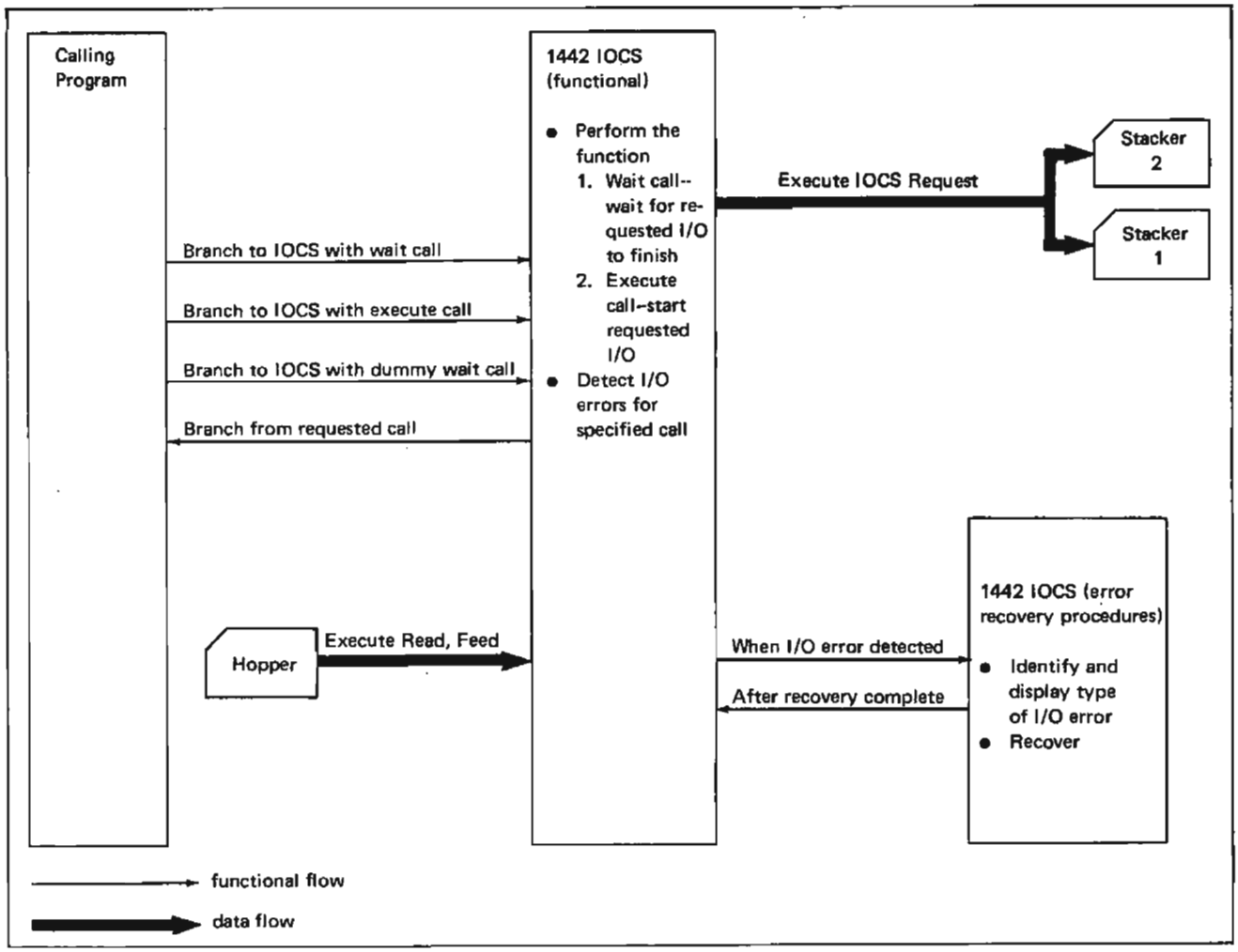


Figure 2-27. Functional and Data Flow for Compiler 1442 IOCS Requests

Compiler

The compiler IOCS routine does not use an IOB or DTF (define the file). The entry point determines the function to be performed. These functions can be requested:

1. Execute call
2. Wait call

DETECT AND RECOVER FROM I/O ERRORS

Object Program

The program checks for I/O errors before determining whether a wait call or execute call was requested. If an error is detected, the IOCS routine branches to the error recovery procedures (ERPs). The ERPs attempt to recover from an I/O error by re-trying as many of the previous operations as necessary to properly complete the function and stack the cards.

Error information is recorded in the Error Recording area for all errors found. A halt code is displayed when the error is detected. For a detailed description of the ERPs, see Section 3, *Program Organization.* Figure 2-29 is a summary of the halt codes for the error types and the program action taken.

The 1442 IOCS recovers from ten types of errors:

- Punch station jam on read command – The halt code is C7. After the cards are replaced in the hopper, a card is fed into the punch station. The last operation is repeated.

- Punch station jam on punch command – The halt code is C8. After the cards are replaced in the hopper, two cards are fed in order to stacker select the card in error. A second halt displays the number of the stacker containing this card. The last operation is repeated.

- Read station jam – The halt code is C6. After the cards are replaced in the hopper, a card is fed into the punch station. The program returns to the IOCS mainline to continue processing.

- Transport jam – The halt code is C9. After the cards are replaced in the hopper, a card is fed into the punch station. If this error occurred prior to the test for a wait call, the program repeats the last operation. If this error occurred during an execute call, the program feeds another card and then repeats the last operation.

- Hopper misfeed – The halt code is C5. After the cards are replaced in the hopper, a card is fed into the punch station. If this error occurred prior to the test for a wait call, the last operation is repeated. If this error occurred during an execute call, the program returns to the IOCS mainline to continue processing.

- Extra feed cycle – The halt code is CA. After the cards are replaced in the hopper, a card is fed into the punch station. The program returns to the IOCS mainline to continue processing.

- Data overrun on punch-feed – The halt code is C3. A second halt displays the number of the stacker containing the card in error. After the cards are replaced in the hopper, the last operation is repeated.

- Data overrun on punch-no feed – The halt code is C4. After the cards are replaced in the hopper, two cards are fed in order to stacker select the card in error. A second halt displays the number of the stacker containing this card. The last operation is repeated.

- Data overrun on read – The halt code is C2. After the cards are replaced in the hopper, the last operation is repeated.

- Read invalid – The halt code is C1. After the cards are replaced in the hopper, the last operation is repeated.

Compiler

The compiler IOCS ERP differs from object program IOCS ERPs in that it does not record error information in the Error Recording area.

Section 3. Program Organization

GENERAL DESCRIPTION OF 1442 IOCS ROUTINES

The overview presented in this section deals with an object program IOCS routine. The compiler IOCS routine differs from object program IOCS routines in these ways:

- Compiler IOCS routine does not use an IOB or DTF.
- Compiler IOCS routine has separate entry points. The possible entry points are:
  1. Execute call
  2. Wait call
  3. Dummy wait call
- Compiler IOCS routine does not record error information in the Error Recording area.

See *1442 IOCS Routines* in this section for flowcharts of specific routines.

Functional Description

The calling program branches to the 1442 IOCS routine to request a 1442 function. Immediately following the branch is a parameter which points to the IOB. The IOB contains the type of call, the function to be performed, the record length, and stacker information. The last two bytes of the IOB point to the DTF. The DTF contains buffer addresses needed by the IOCS routines and provides space for storage of error information (see Section 4, *Data Area Formats*, for the contents of the IOB and DTF). Figure 2-28 shows the linkage of these areas.

The 1442 IOCS routine waits for all previous operations on the 1442 to complete and then checks for errors. If an error is detected, the program branches to the ERPs. If no errors have occurred, IOCS checks the IOB to determine whether a wait call or an execute call is requested.

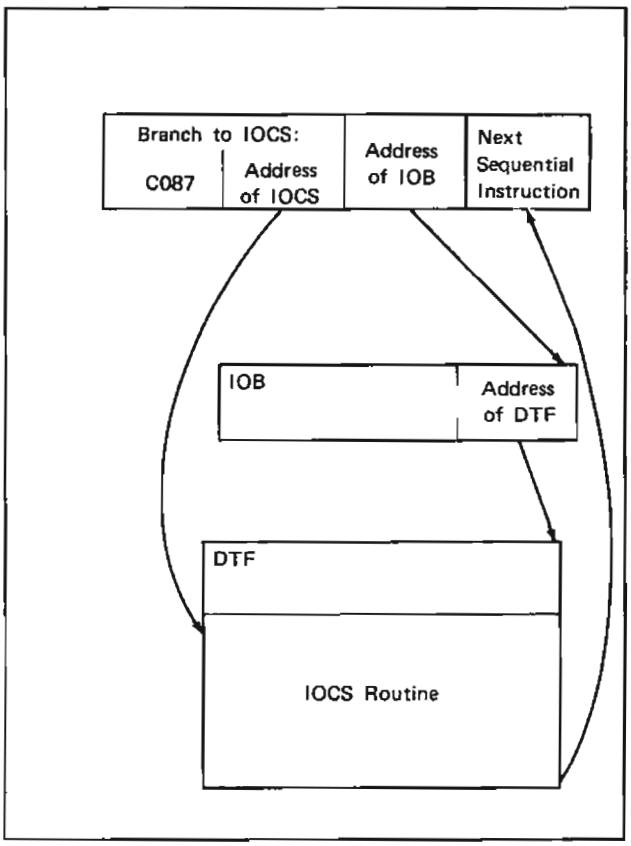


Figure 2-28. Linkage for Object Program 1442 IOCS Routines

Halt Code	Condition	Priority	Status		I/O Attention Light	Program Action
			Byte	Bit		
None	Simple not ready	8	1	4	On	Continue processing.
C7	Punch station jam on read command	1	1 2	6 6	Off	Feed one card and reissue the last command.
C8	Punch station jam on punch command	1	1 2	6 6	Off	Feed two cards and reissue the last command.
C6	Read station jam	2	1 2	6 3	Off	Feed one card and return to mainline to continue processing.
C9	Transport jam	3	1 2	6 7	Off	Execute call--feed two cards and reissue the last command; wait call--feed one card and reissue the last command.
C5	Hopper misfeed	4	1 2	6 4	Off	Execute call--feed one card and reissue the last command; wait call--feed one card and return to mainline to continue processing.
CA	Extra feed cycle	5	1 2	6 5	Off	Feed one card and return to mainline to continue processing.
C3	Data overrun on punch-feed	6	1 1	2 3	Off	Reissue the last command.
C4	Data overrun on punch-no feed	6	1 1	2 3	Off	Feed two cards and reissue the last command.
C2	Data overrun on read	6	1 1	0 3	Off	Reissue the last command.
C1	Read invalid	7	1	7	Off	Reissue the last command.
None	No-op	—	1	5	Off	————

Note 1: Errors are processed in the order they appear on this chart.

Note 2: Byte 1 refers to the second byte (high storage address) of sense data.

Figure 2-29. Error Recovery Information for 1442 IOCS Routines

Wait Call

If a wait call is requested, IOCS checks to see if a punch wait is specified. If it is not, IOCS returns to the calling program. If it is, IOCS clears the punch buffer to blanks and then returns to the calling program. Chart IP, part 1, shows an overview of the wait call. For a more detailed flowchart, see *Detailed Description of the 1442 IOCS Routines* in this section.

Execute Call

If an execute call is requested, IOCS performs the following steps to complete the functions:

- Builds the SIO instruction.
- Determines if the record length specified is equal to 0 or greater than 80. If it is, the length defaults to 80.
- Initiates the I/O operation (SIO).

When the I/O operation has been completed for a wait call or accepted for an execute call, IOCS returns to the calling program. Chart IP, part 1, shows an overview of the execute call. For a more detailed flowchart, see *Detailed Description of the 1442 IOCS Routines* in this section.

Error Recovery Procedures Description

The program branches to the error recovery procedures when it detects the device not ready. The status bytes are sensed into the DTF and loaded into the ARR (address recall register) from the DTF. Figure 2-29 shows the status sense information for each error.

A check is then made for I/O errors. If none have occurred, the error is a simple device-not-ready; the IOCS routine continues processing.

The Q, R, and SNS bytes in the DTF are moved into the Error Recording area for all errors except a simple device-not-ready. See Section 4, *Data Area Formats*, for a description of the Error Recording area.

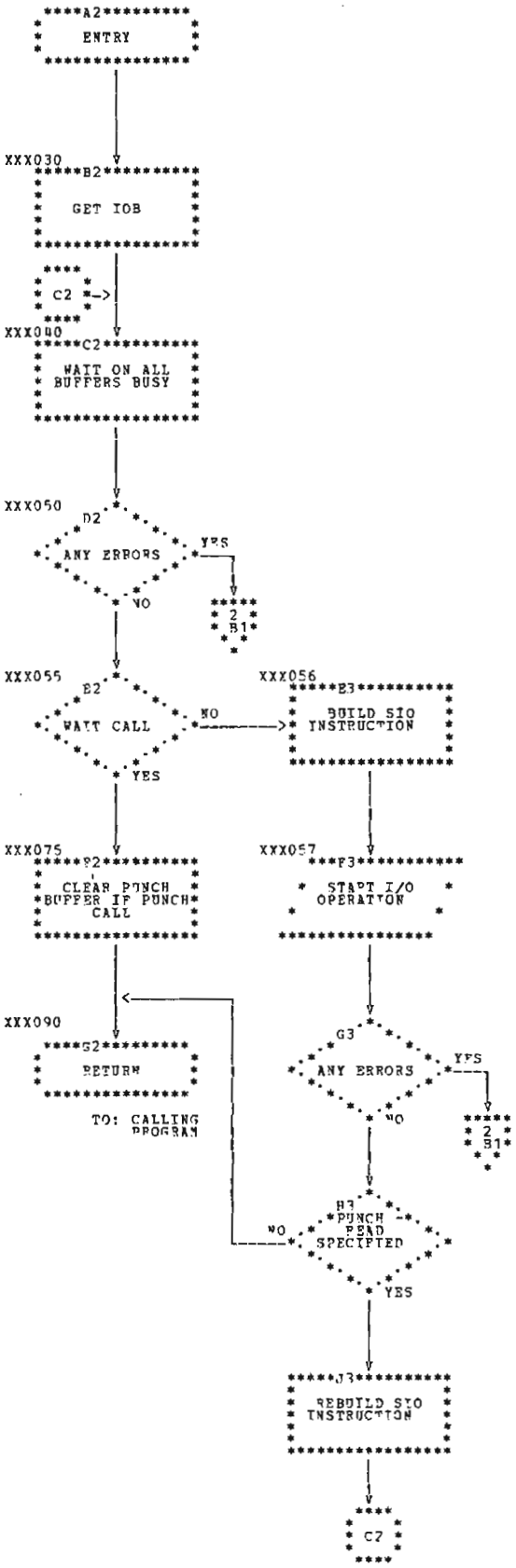


Chart IP. Overview–Wait and Execute Calls (Part 1 of 2)

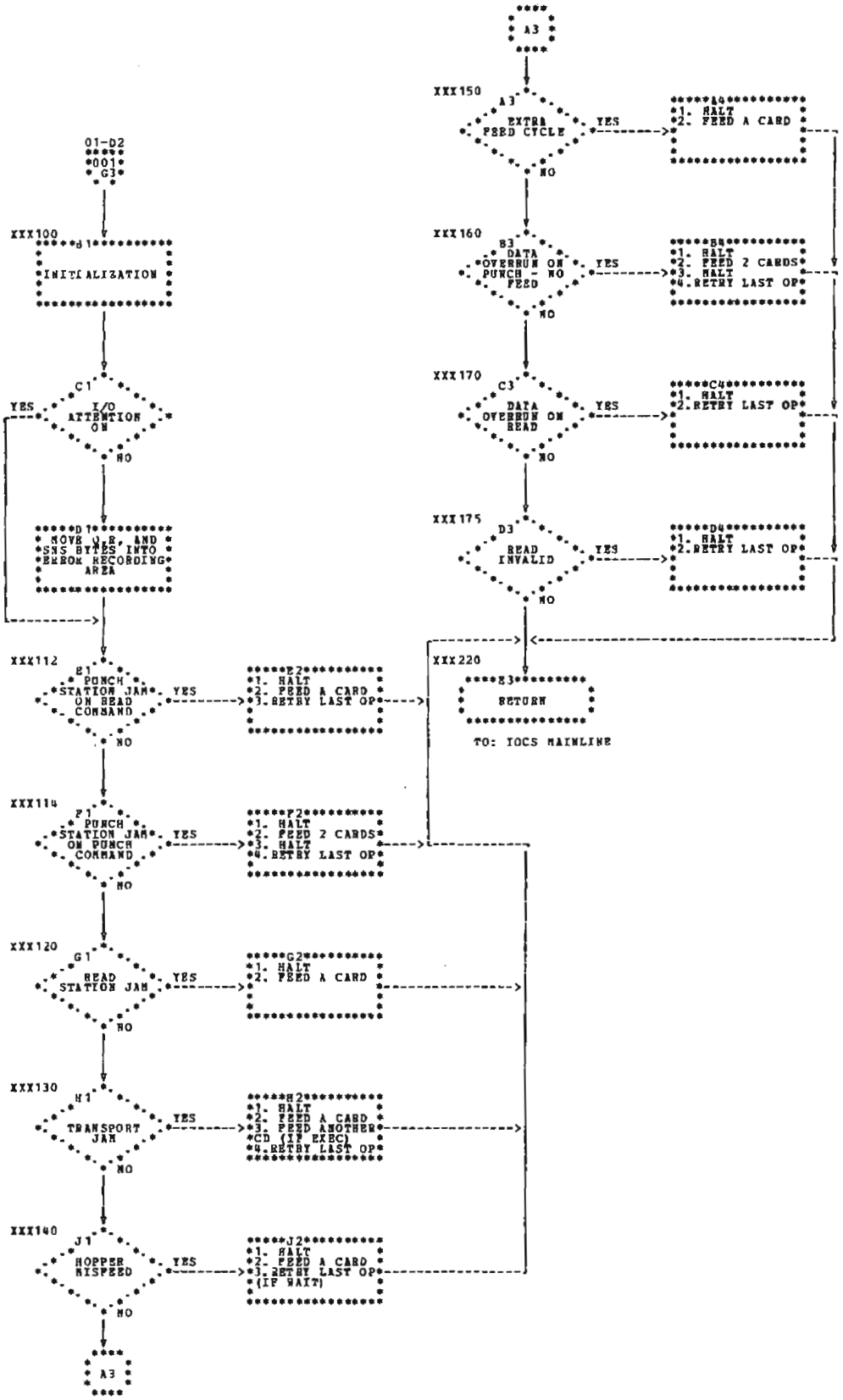


Chart IP. Overview–Error Recovery Procedures (Part 2 of 2)

A counter is set prior to each error check which indicates the number of cards to be fed for error recovery of that particular error. Ten types of errors are checked for in this order:

1. Punch station jam on read command
2. Punch station jam on punch command
3. Read station jam
4. Transport jam
5. Hopper misfeed
6. Extra feed cycle
7. Data overrun on punch-no feed
8. Data overrun on punch-feed
9. Data overrun on read
10. Read invalid

Chart 1P, part 2, shows an overview of the ERPs. For a more detailed flowchart, see *Detailed Description of the 1442 IOCS routines* in this section.

Punch Station Jam on Read

The halt code for a punch station jam on read is displayed (C7). The program issues one feed command and reissues the instruction on which the error occurred. Figure 2-30 shows a schematic diagram of the punch station jam on read ERP.

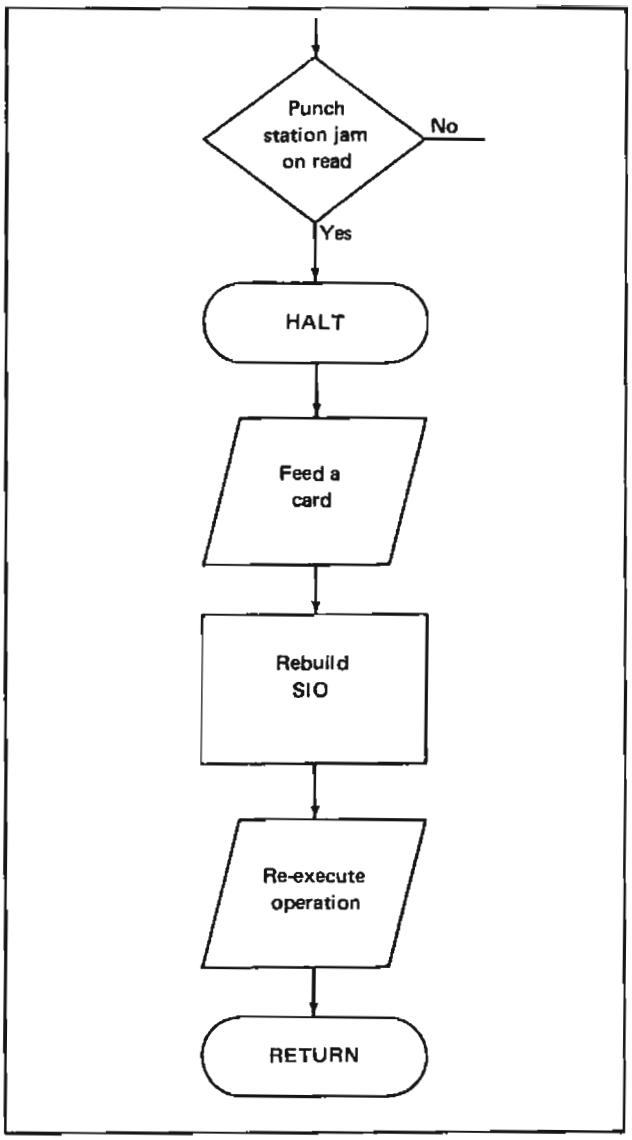


Figure 2-30. Schematic Diagram of the Punch Station Jam on Read ERP

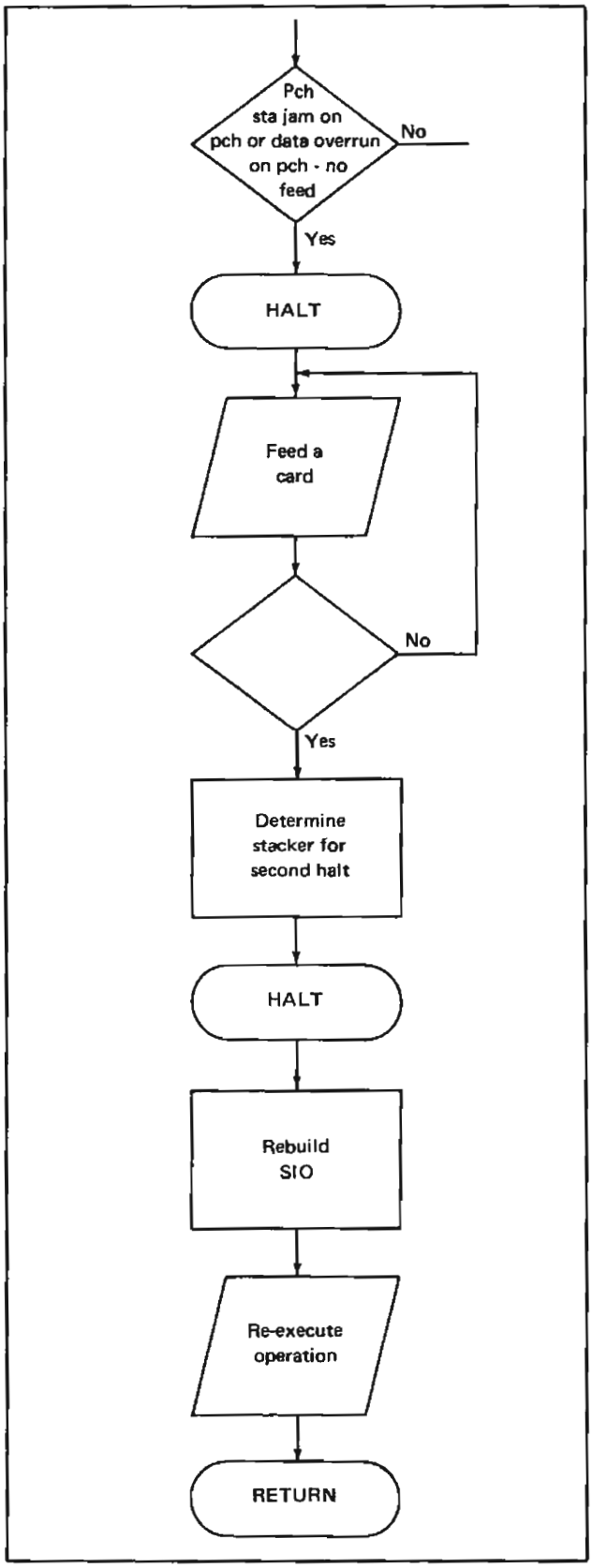


Figure 2-31. Schematic Diagram of the Punch Station Jam on Punch or Data Overrun on Punch-No Feed ERP

Punch Station Jam on Punch or Data Overrun on Punch-No Feed

Error recovery is the same for these two errors. The correct halt code is displayed (C8 or C4). The program issues two feed commands and determines the stacker which contains the card in error. The second halt is displayed. The program then reissues the instruction on which the error occurred. Figure 2-31 shows a schematic diagram of the punch station jam on punch and data overrun on punch-no feed ERP.

Read Station Jam or Extra Feed Cycle

Error recovery is the same for these two errors. The correct halt code is displayed (C6 or CA). The program issues one feed command and returns to the IOCS mainline to continue processing. Figure 2-32 shows schematic diagram of the punch station jam and extra feed cycle ERP.

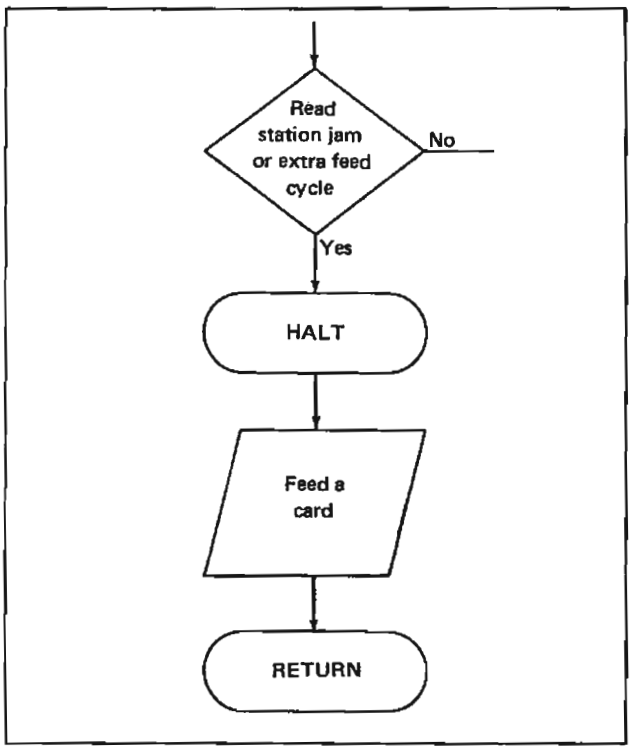


Figure 2-32. Schematic Diagram of the Read Station Jam or Extra Feed Cycle ERP

Transport Jam

The halt code for a transport jam is displayed (C9). If an execute call has been specified, the program issues two feed commands and reissues the instruction on which the error occurred. If a wait call has been specified, the program issues one feed command and reissues the instruction on which the error occurred. Figure 2-33 shows a schematic diagram of the transport jam ERP.

Hopper Misfeed

The halt code for a hopper misfeed is displayed (C5). The program issues one feed command. If an execute call has been specified, the program then returns to the IOCS mainline to continue processing. If a wait call has been specified, the program reissues the instruction on which the error occurred. Figure 2-34 shows a schematic diagram of the hopper misfeed ERP.

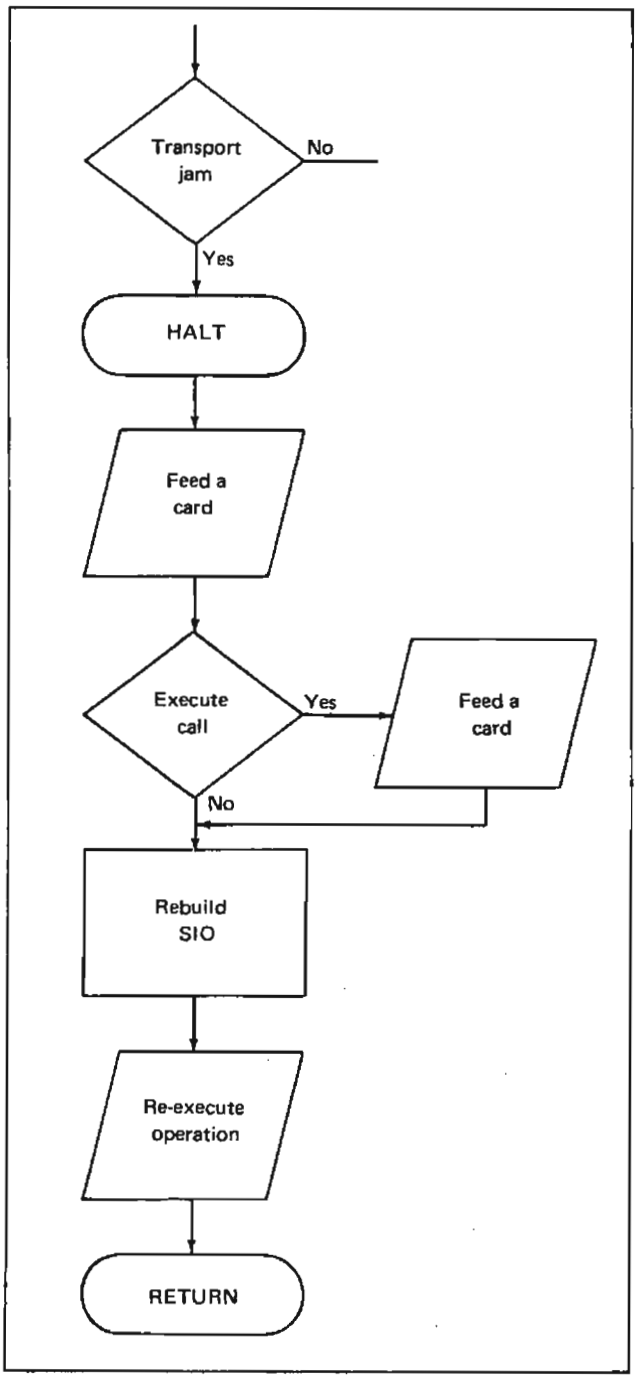


Figure 2-33. Schematic Diagram of the Transport Jam ERP

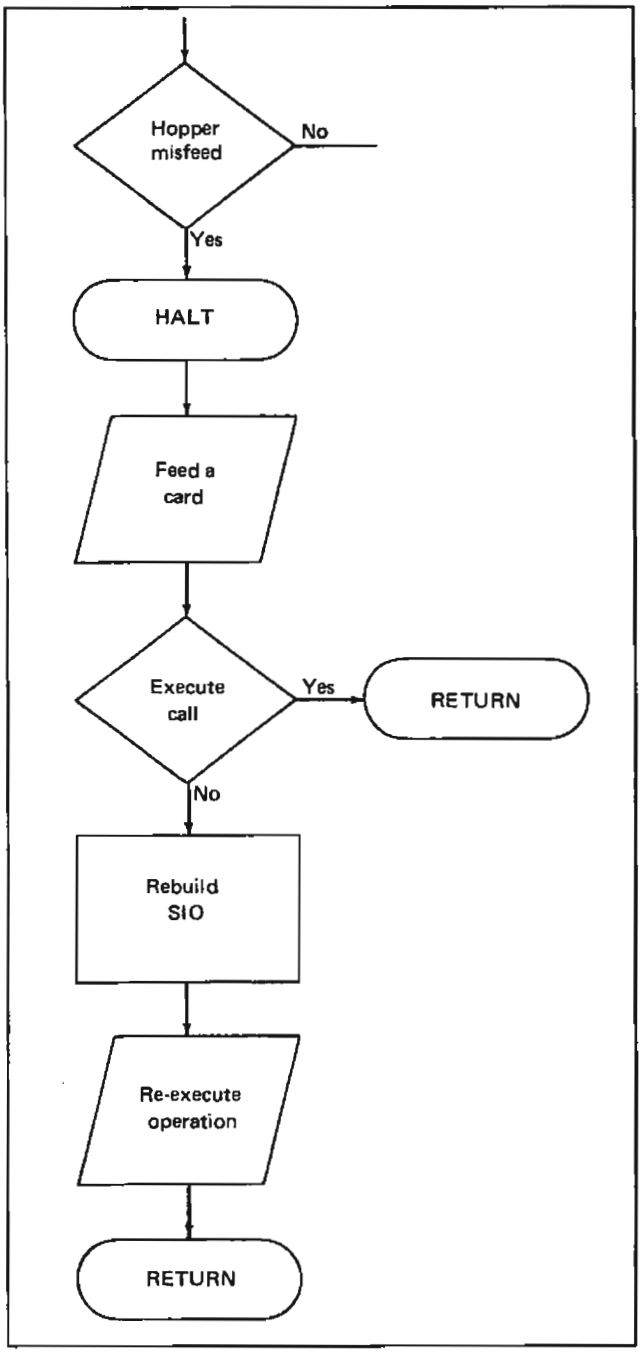


Figure 2-34. Schematic Diagram of the Hopper Misfeed ERP

Data Overrun on Punch-Feed

The halt code for a data overrun on punch-feed is displayed (C3). The program determines the stacker which contains the card in error and displays the second halt. The instruction on which the error occurred is reissued. Figure 2-35 shows a schematic diagram of the data overrun on punch-feed ERP.

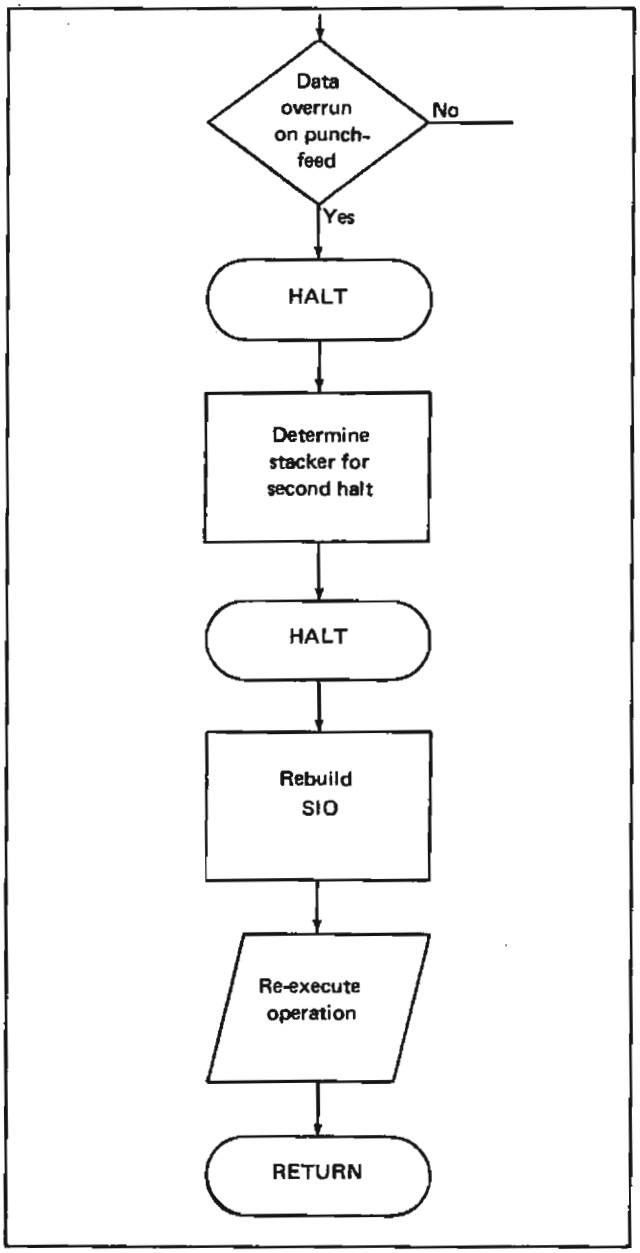


Figure 2-35. Schematic Diagram of the Data Overrun on Punch-Feed ERP

Read Invalid or Data Overrun on Read

Error recovery is the same for these two errors. The correct halt code is displayed (C1 or C2). The program reissues the instruction on which the error occurred. Figure 2-36 shows a schematic diagram of the read invalid or data overrun on read ERP.

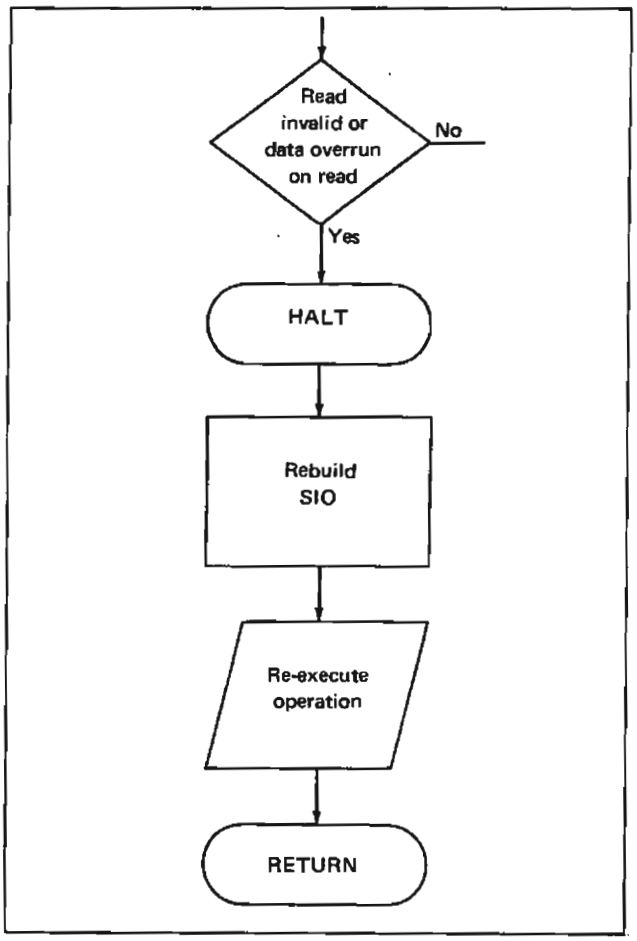


Figure 2-36. Schematic Diagram of the Read Invalid or Data Overrun on Read ERP

DETAILED DESCRIPTION OF THE 1442 IOCS ROUTINES

This section contains detailed flowcharts of the overview flowchart presented previously (Chart IP). Chart IQ is broken down in this manner:

- Part 1 – Initialization and wait call
- Part 2 – Execute call
- Parts 3 through 6 – Error recovery procedures

Notes relating to specific blocks in the flowcharts are included for more thorough explanations.

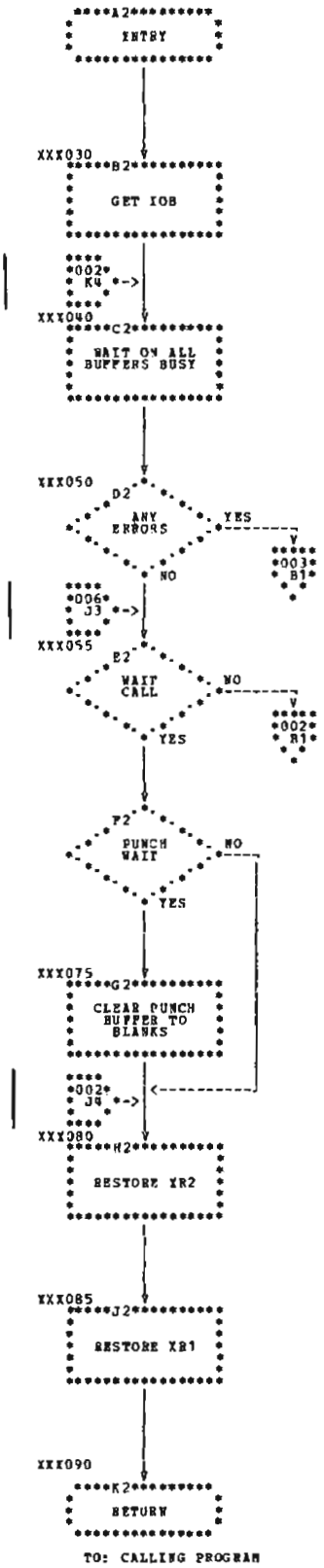
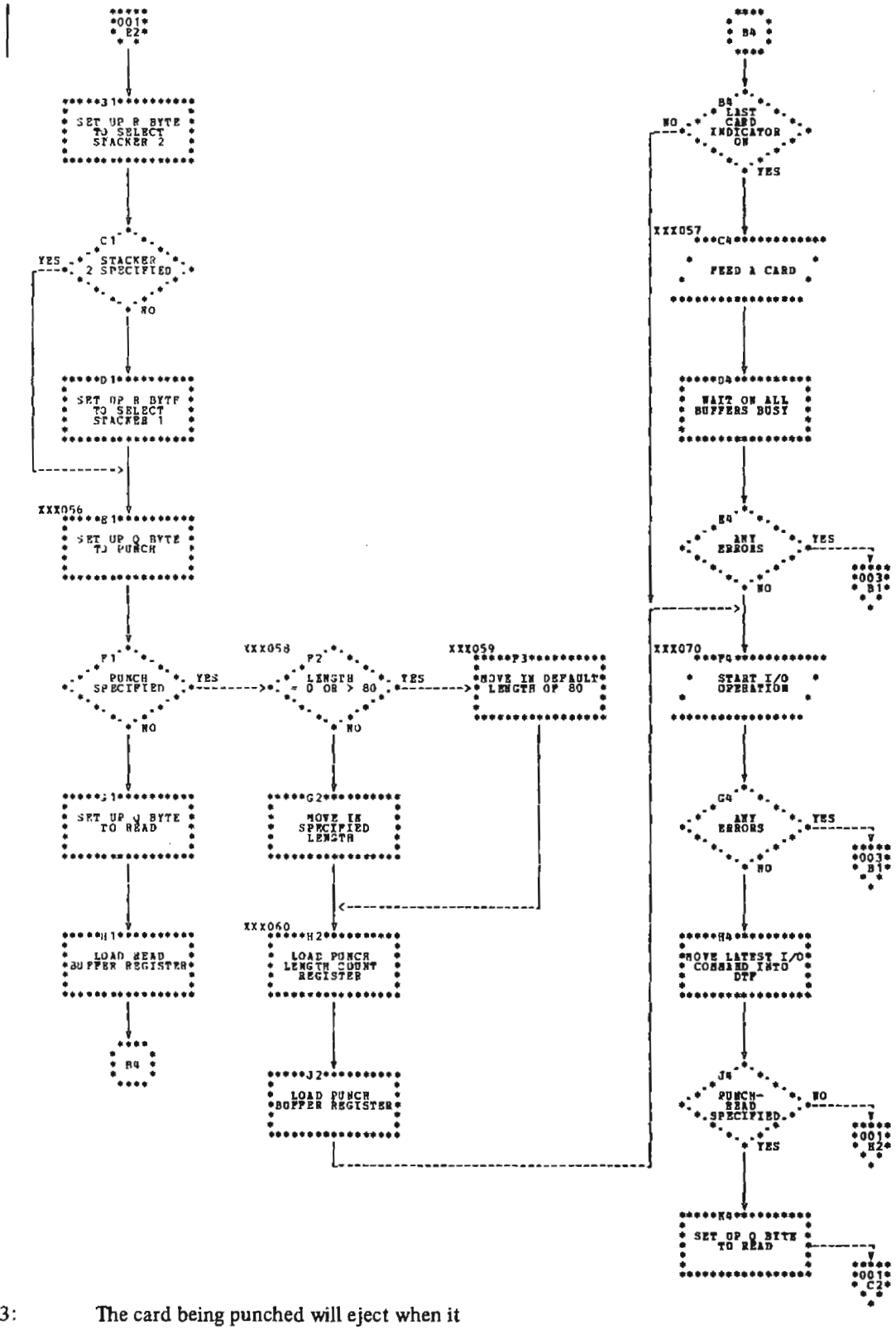


Chart IQ. Initialization and Wait Call (Part 1 of 6)



02F3: The card being punched will eject when it reaches the specified column. Therefore, in order to increase card throughput, it is better to specify the record length than to allow the length to default.

Chart IQ. Execute Call (Part 2 of 6)

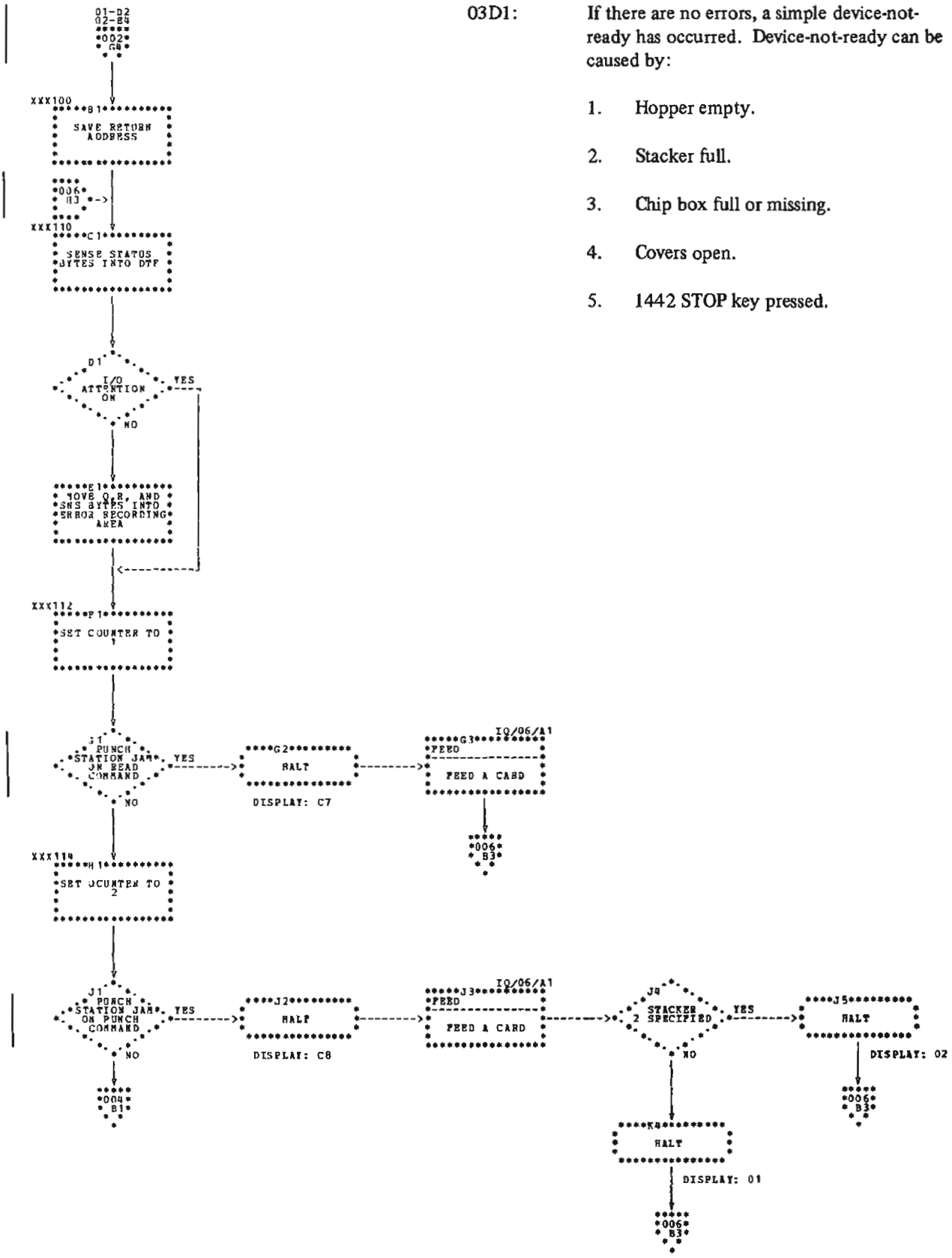


Chart IQ. Error Recovery Procedures (Part 3 of 6)

03D1: If there are no errors, a simple device-not-ready has occurred. Device-not-ready can be caused by:

1. Hopper empty.
2. Stacker full.
3. Chip box full or missing.
4. Covers open.
5. 1442 STOP key pressed.

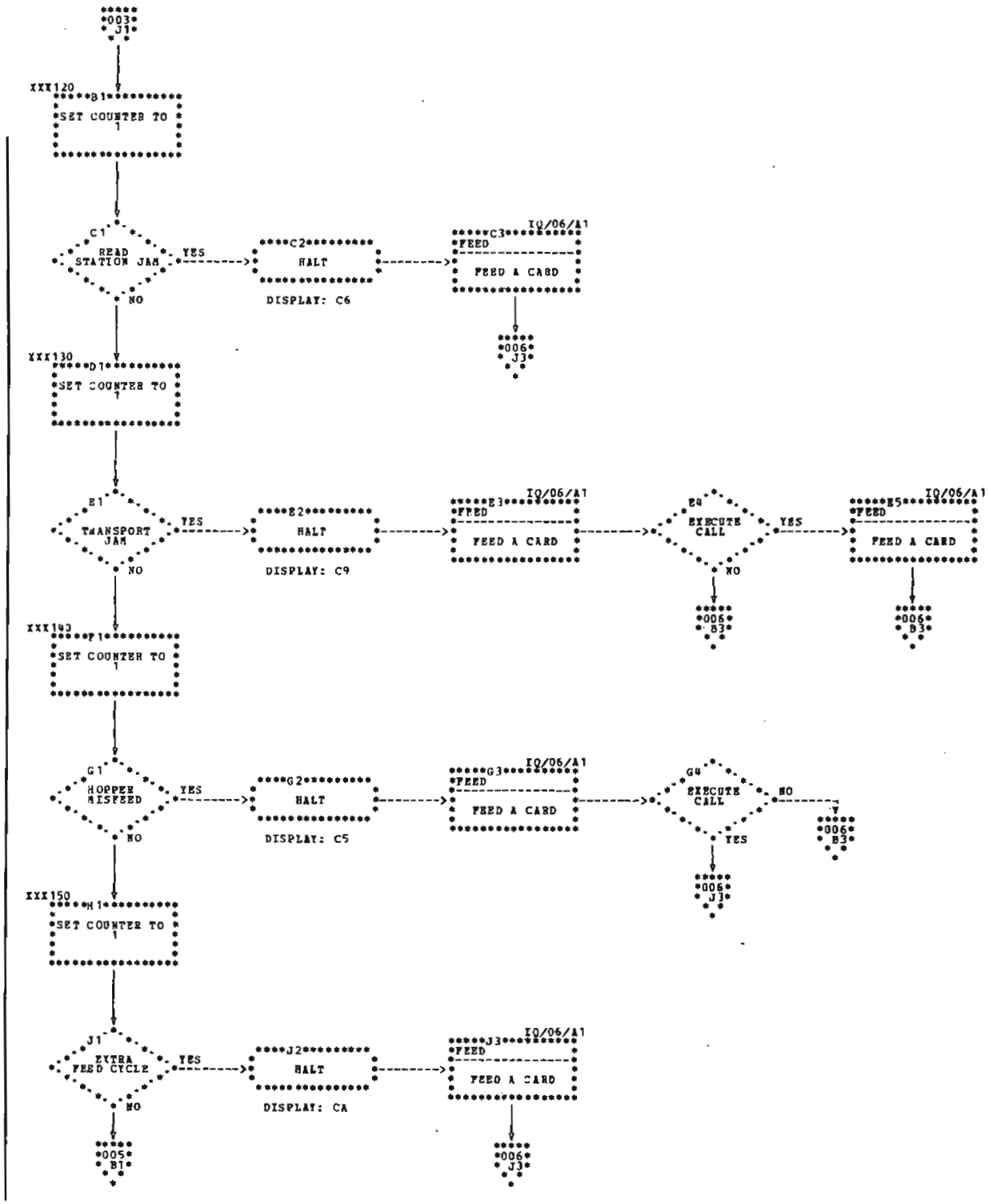


Chart IQ. Error Recovery Procedures (Part 4 of 6)

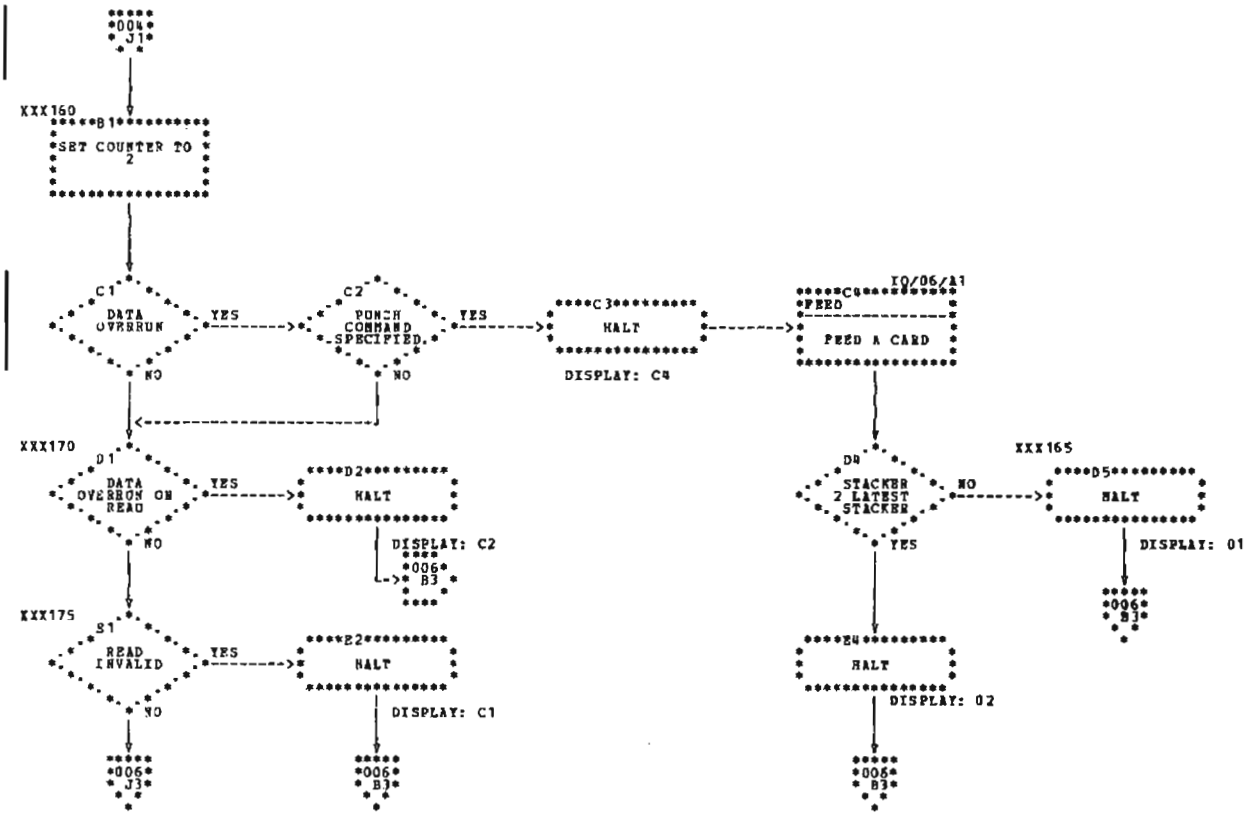
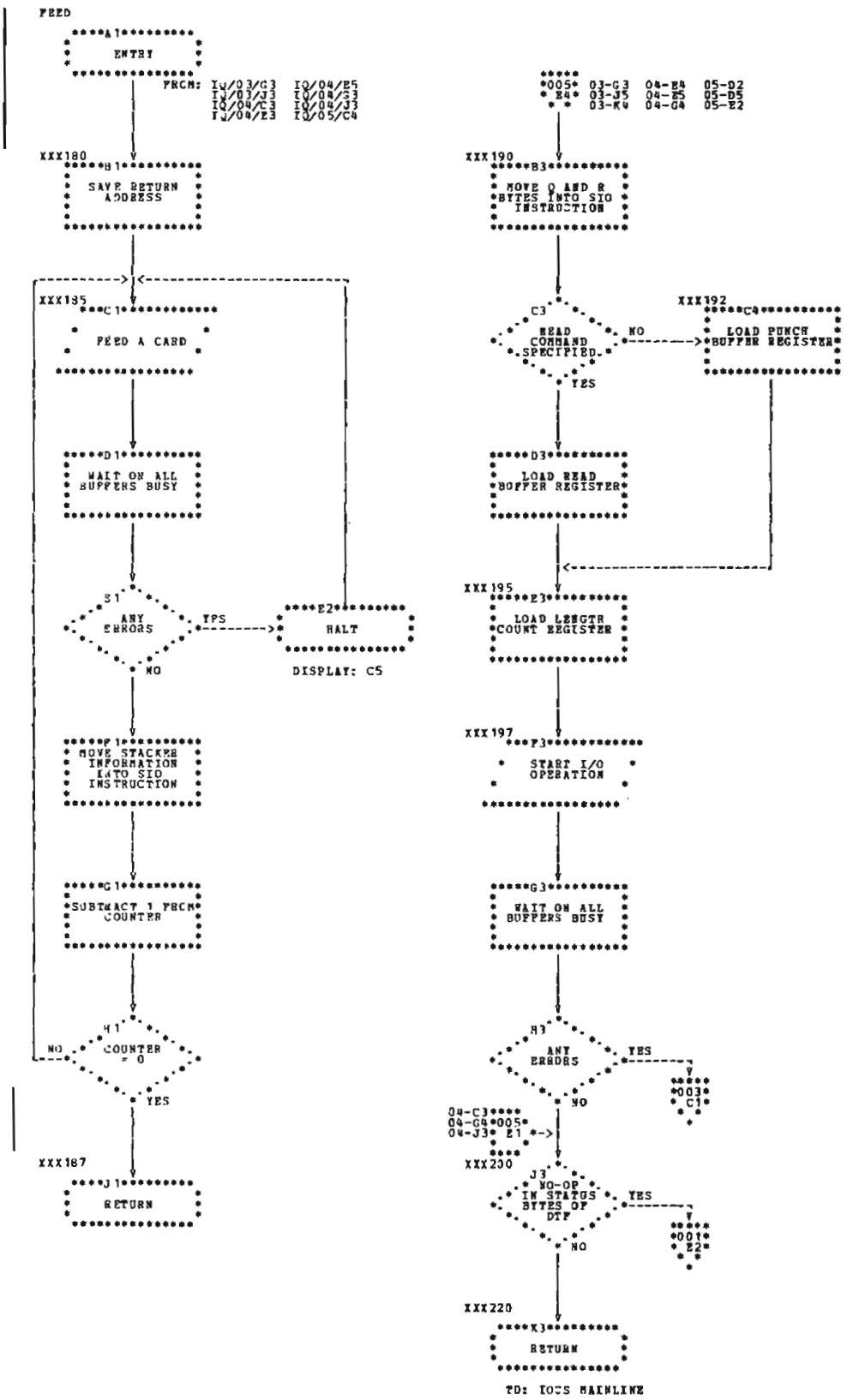


Chart IQ. Error Recovery Procedures (Part 5 of 6)



06J3: This block detects hopper misfeed during the hardware prime.

Chart IQ. Error Recovery Procedures (Part 6 of 6)

1442 IOCS ROUTINES

This section gives a detailed description of each 1442 IOCS routine. The functions of each routine are given as well as its entry points, input required to perform the functions, resulting output, and exits. Distinctions from the general IOCS routine previously described are indicated. The flowcharts for a routine are on level similar to Chart IP in *General Description of 1442 IOCS Routines.* For a more detailed flowchart and discussion, see Chart IQ.

Read/Punch-No Feed (Object)

Entry Point: ARGEZ1

Chart: IR

Functions:

- Waits for the I/O operation to complete.
- Reads cards from the hopper.
- Punches cards from the hopper.
- Provides stacker selection.
- Detects and recovers from errors.

Input:

- IOB
- DTF

Output: Punched 80-column cards

Exit: Control is returned to the instruction immediately following the 2-byte parameter pointing to the IOB.

Routine Distinctions:

- DTF is 8 bytes.
- All ERPs are provided except data overrun on punch-feed.

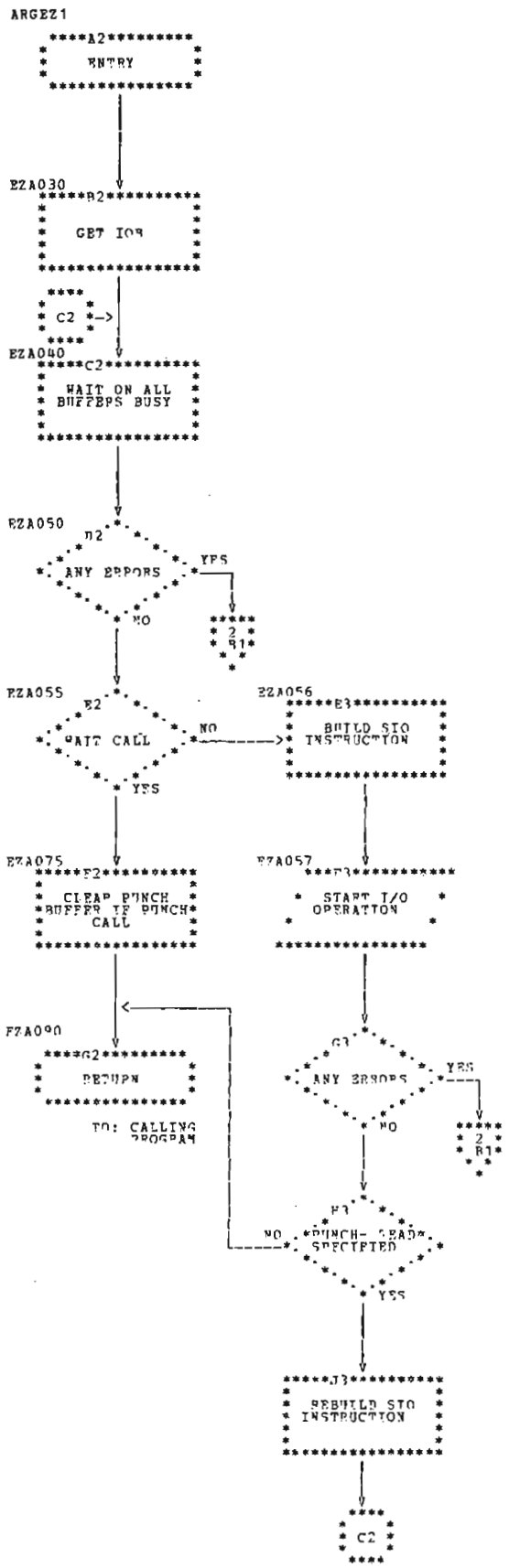


Chart IR. Read/Punch-No Feed (Object)–Wait and Execute Calls (Part 1 of 2)

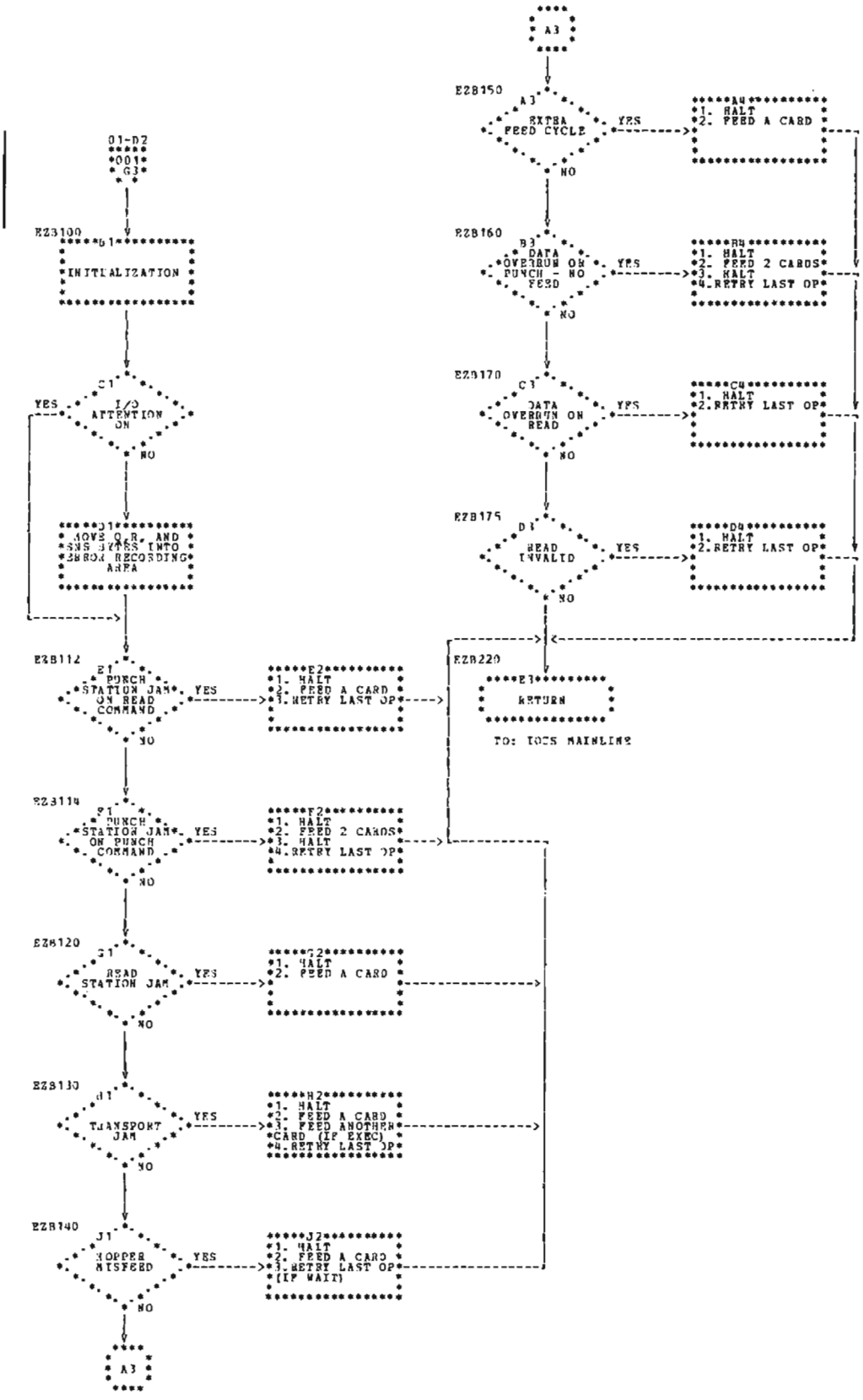


Chart IR. Read/Punch-No Feed (Object)–Error Recovery Procedures (Part 2 of 2)

Read Column Binary (Object)

Entry Point: ARGFA1

Chart: IS

Functions:

- Waits for the I/O operation to complete.
- Reads cards in column binary format.
- Provides stacker selection.
- Detects and recovers from errors.

Input:

- IOB
- DTF

Output: None

Exit: Control is returned to the instruction immediately following the 2-byte parameter pointing to the IOB.

Routine Distinctions:

- Read buffer is 160 bytes.
- DTF is 6 bytes.
- All ERPs are provided except:
  1. Read invalid.
  2. Data overrun on punch-feed.
  3. Data overrun on punch-no feed.
  4. Punch station jam on punch command.

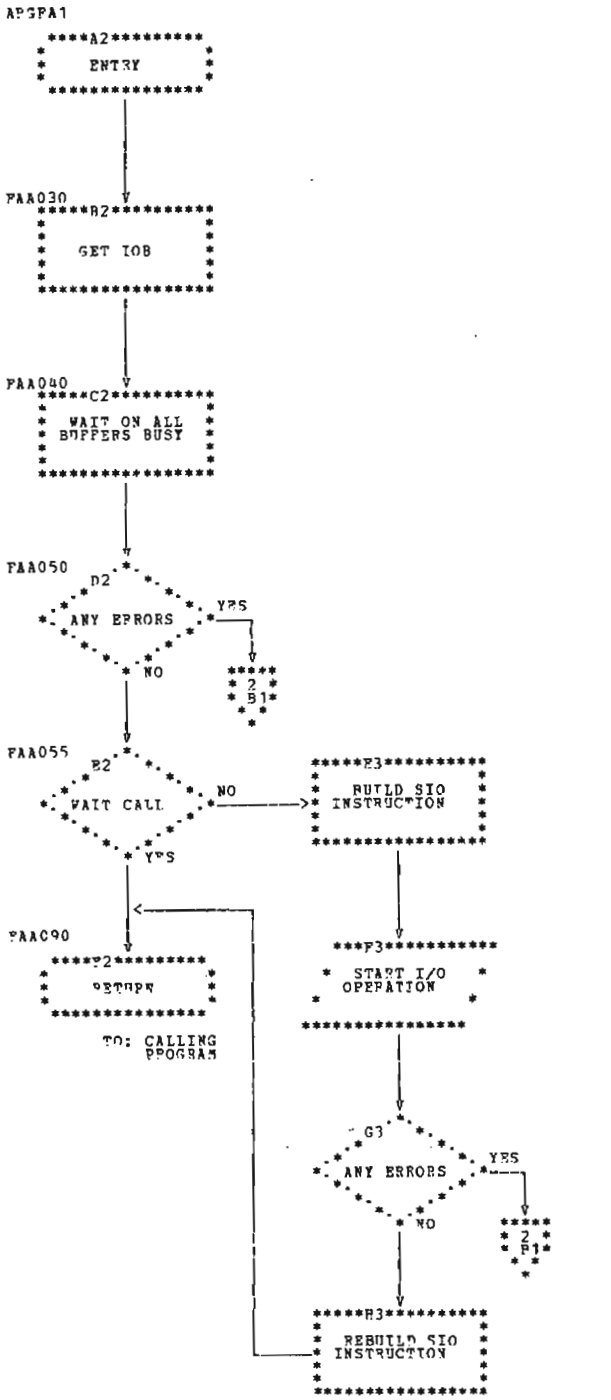


Chart IS. Read Column Binary (Object)—Wait and Execute Calls (Part 1 of 2)

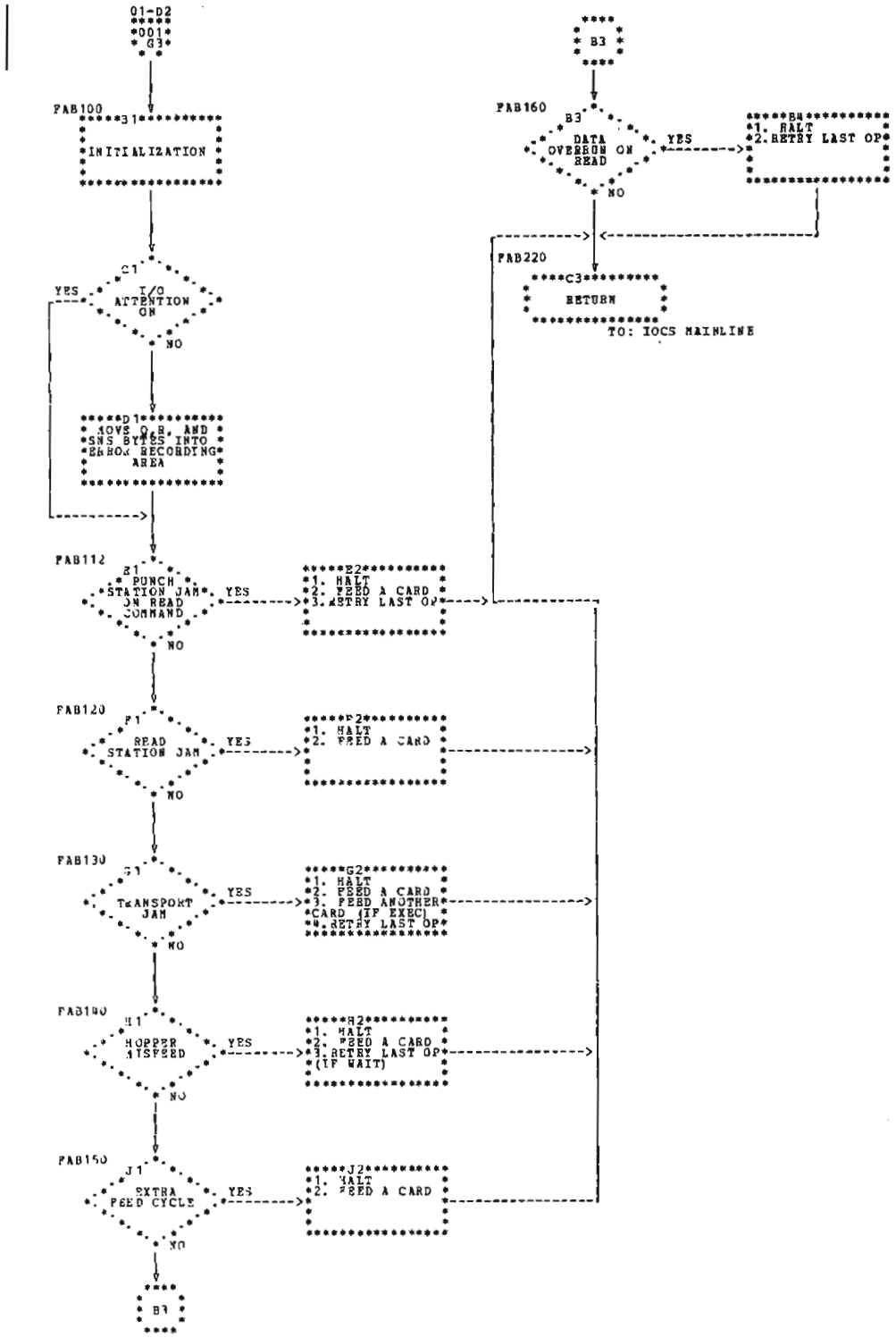


Chart IS. Read Column Binary (Object)–Error Recovery Procedures (Part 2 of 2)

Read Only (Object)

Entry Point: ARGEX1

Chart: IT

Functions:

- Waits for the I/O operation to complete.
- Reads cards from the hopper.
- Provides stacker selection.
- Detects and recovers from errors.

Input:

- IOB
- DTF

Output: None

Exit: Control is returned to the instruction immediately following the 2-byte parameter pointing to the IOB.

Routine Distinctions:

- DTF is 6 bytes.
- All ERPs are provided except:
  1. Data overrun on punch-feed.
  2. Data overrun on punch-no feed.
  3. Punch station jam on punch command.

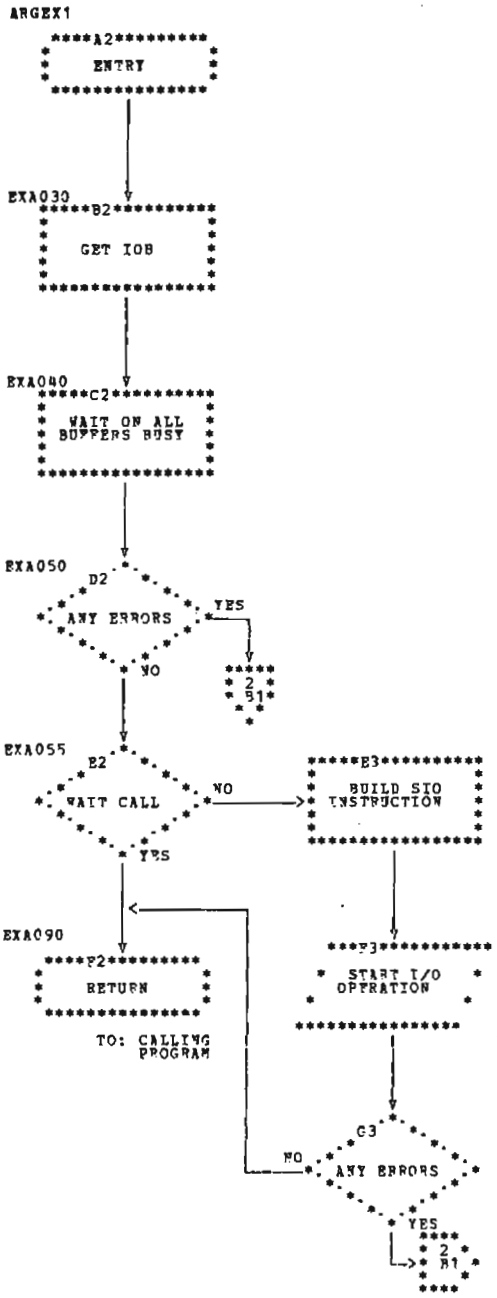


Chart IT. Read Only (Object)–Wait and Execute Calls (Part 1 of 2)

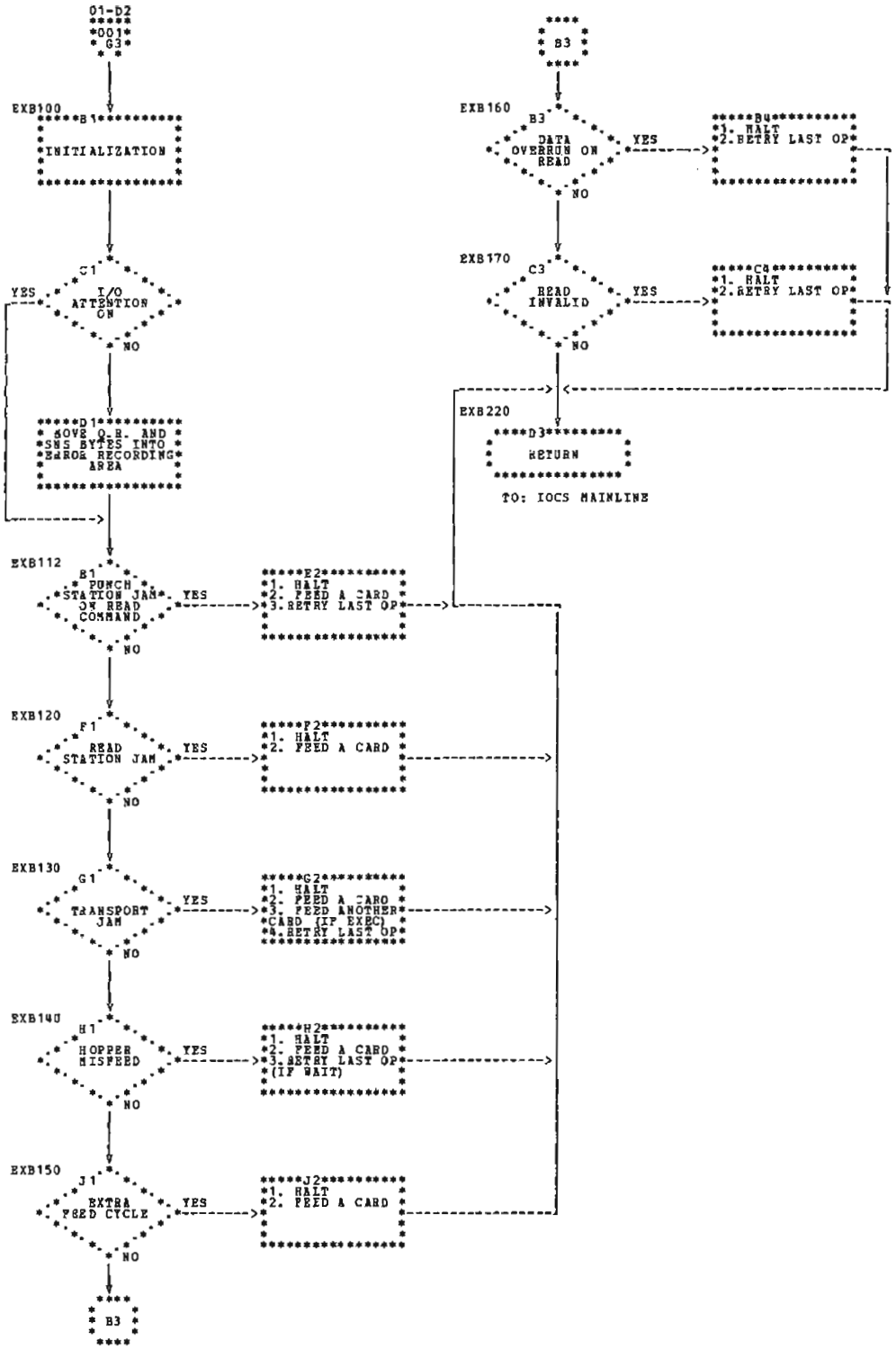


Chart IT. Read Only (Object)–Error Recovery Procedures (Part 2 of 2)

Entry Points: This routine, if used, overlays the Read Only/Secondary Hopper MFCU IOCS (Compiler). Entry point ARGFC2 in this routine is included to replace the entry to wait on cards to clear transport in the MFCU IOCS routine.

- ARGFC1 – Execute call
- ARGFC2 – Dummy wait call
- ARGFC3 – Wait call

Chart: IU

Functions:

- Waits for the I/O operation to complete.
- Reads cards from the hopper.
- Directs cards to stacker 1.
- Detects and recovers from errors.

Input: None

Output: None

Exit: Control is returned to the instruction immediately following the branch to this routine.

Routine Distinctions: All ERPs are provided except:

1. Data overrun on punch-feed.
2. Data overrun on punch-no feed.
3. Punch station jam on punch command.

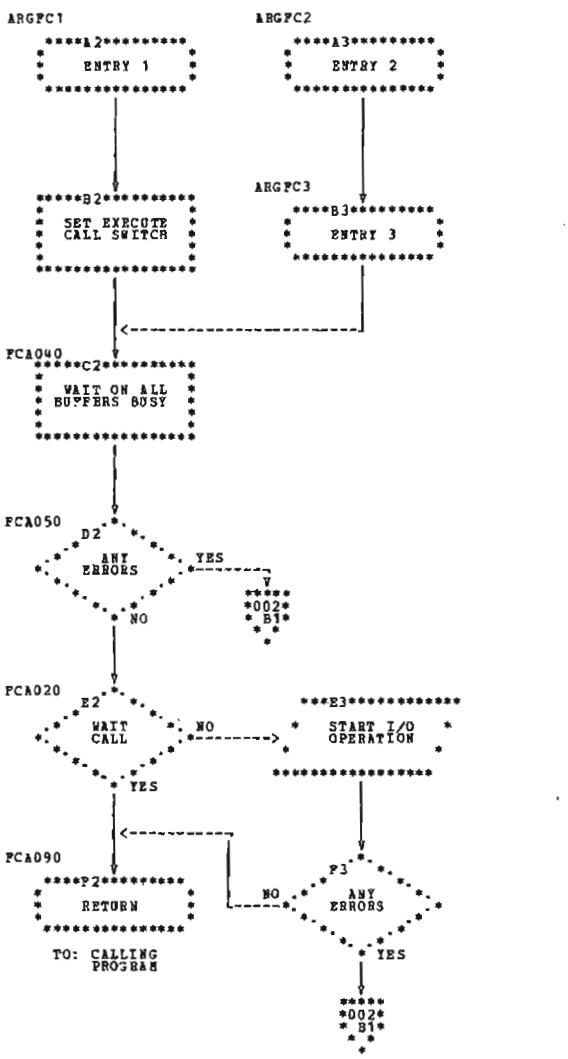


Chart IU. Read Only (Compiler)–Wait and Execute Calls (Part 1 of 2)

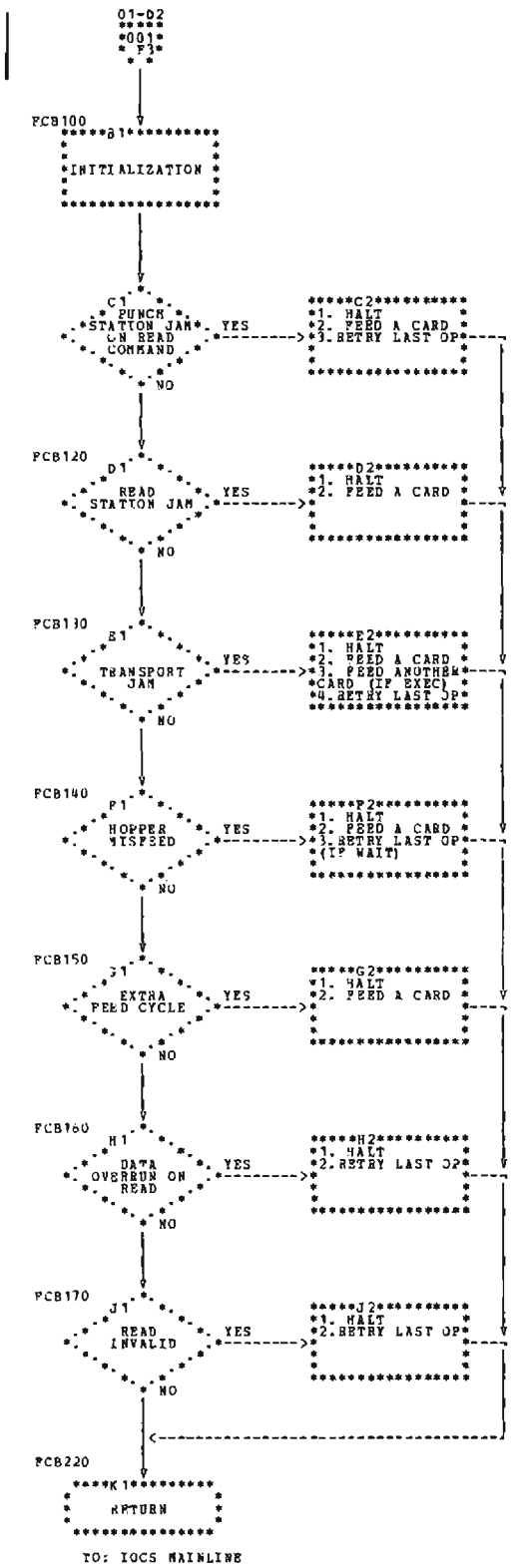


Chart IU. Read Only (Compiler)–Error Recovery Procedures (Part 2 of 2)

Punch-Feed (Object)

Entry Point: ARGEY1

Chart: IV

Functions:

- Waits for the I/O operation to complete.
- Punches cards from the hopper.
- Provides stacker selection.
- Detects and recovers from errors.

Input:

- IOB
- DTF

Output: Punched 80-column cards

Exit: Control is returned to the instruction immediately following the 2-byte parameter pointing to the IOB.

Routine Distinctions:

- DTF is 6 bytes.
- All ERPs are provided except:
  1. Read invalid.
  2. Data overrun on read.
  3. Data overrun on punch-no feed.
  4. Punch station jam on read command.

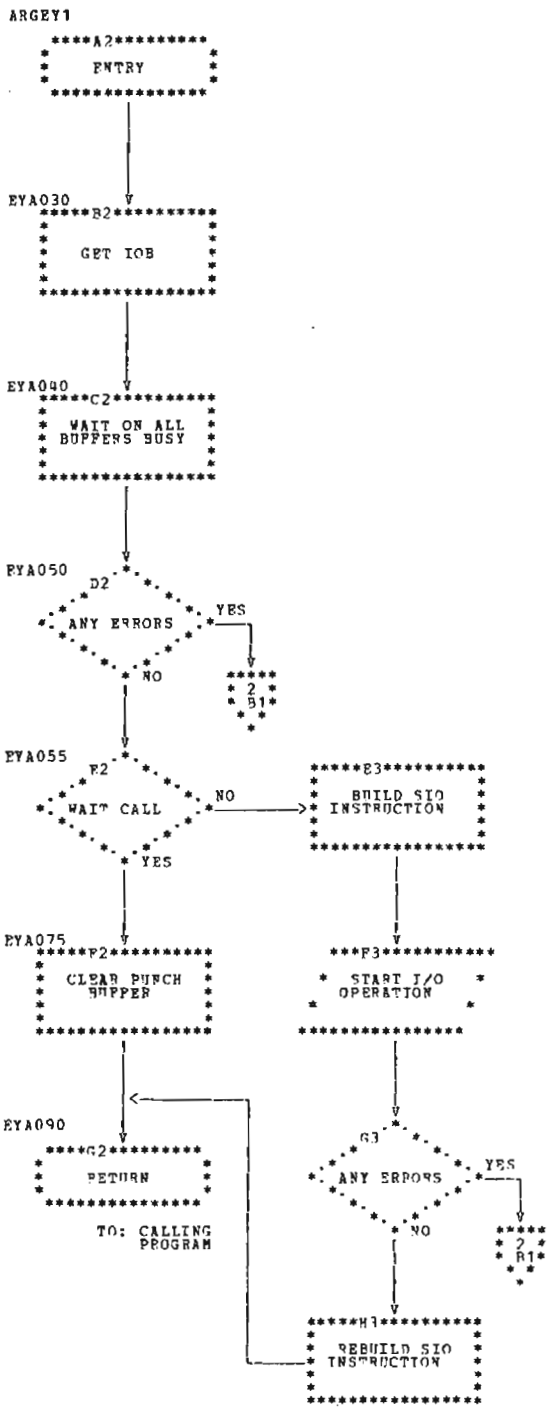


Chart IV. Punch-Feed (Object)—Wait and Execute Calls (Part 1 of 2)

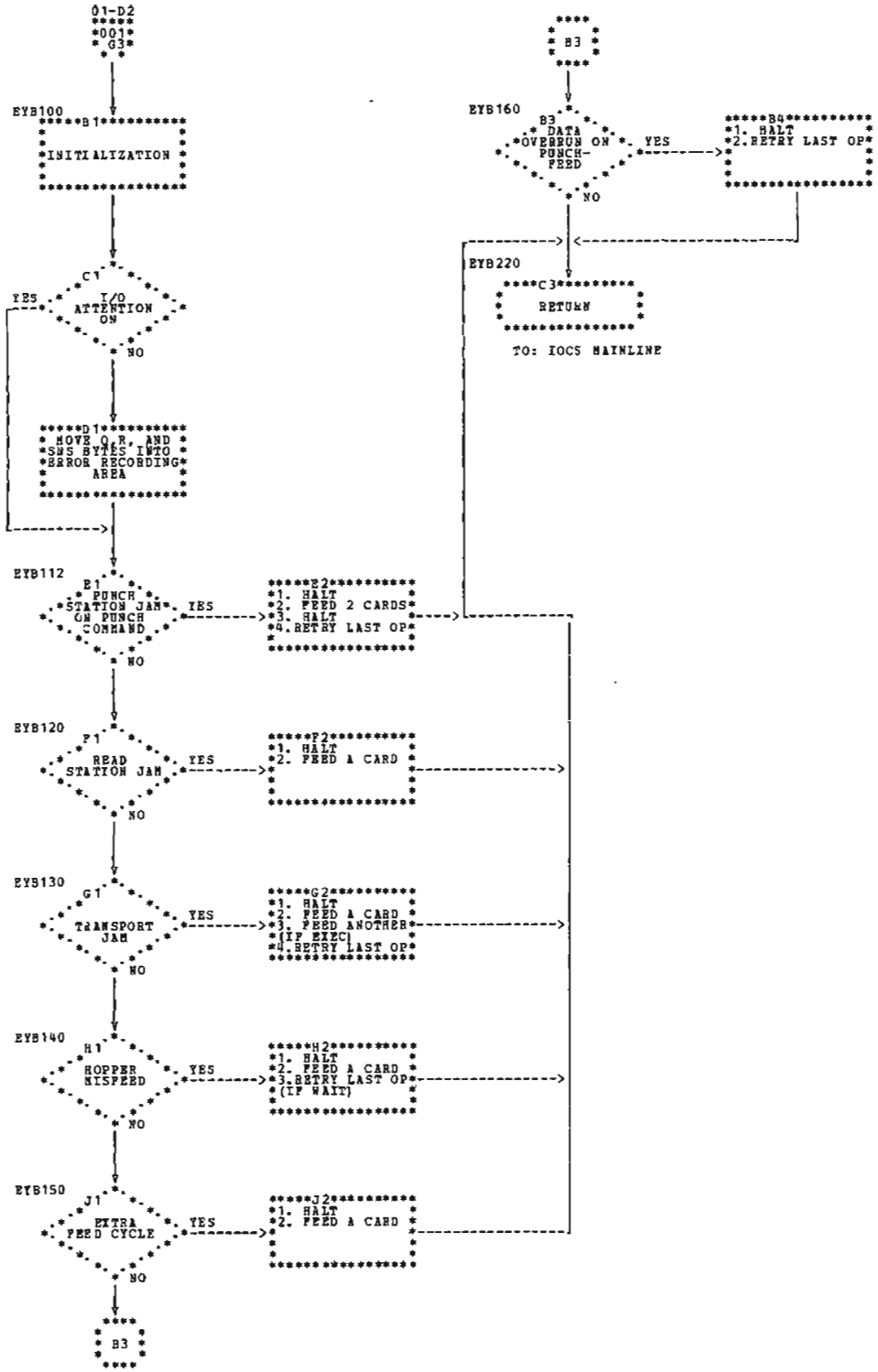


Chart IV. Punch-Feed (Object)–Error Recovery Procedures (Part 2 of 2)

IOB (INPUT/OUTPUT BLOCK)

Information is passed between object programs and the 1442 IOCS routines with a 6-byte IOB. A 2-byte parameter immediately following the branch to the 1442 IOCS routine points to the low-order byte of the IOB. The contents of the IOB are shown in Figure 2-37.

DTF (DEFINE THE FILE)

The last two bytes of the IOB point to a DTF. The DTF contains buffer addresses and error information. The contents of the DTF, depending on the 1442 IOCS routine called, are shown in Figure 2-38.

ERROR RECORDING AREA

This 42-byte area starts at location X'180'. This area, a history table of I/O device errors, 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) to the true print position. See the System Initialization program for more information.

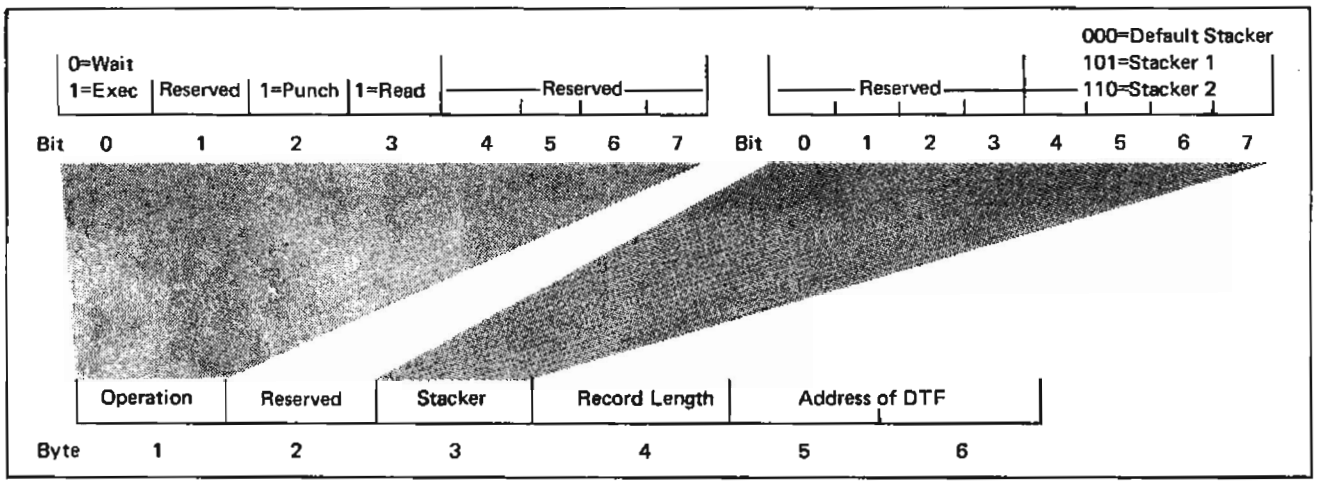


Figure 2-37. IOB Information for 1442 IOCS Routines

Byte / Phase	1	2	3	4	5	6	7	8
1442 Read/ Punch-No Feed	Address of read buffer		Address of punch buffer		Error information			
					Q	R	Status sense	
1442 Read Column Binary	Address of read buffer		Error information					
			Q	R	Status sense			
1442 Read Only	Address of read buffer		Error information					
			Q	R	Status sense			
1442 Punch-Feed	Address of punch buffer		Error information					
			Q	R	Status sense			

Figure 2-38. DTF Information for 1442 IOCS Routines

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

Descriptive Name	*Chart*	*Entry Point*	*Function*
Absolute Card Loader	AA	ALDAA1	• Reads object program cards • Moves data to a storage location • Branches to the program entry point
Full Function MFCU IOCS (Object)	IC	ARGEM1	• Reads, punches, and prints cards from both hoppers • Provides error recovery
Full Function Modified for Data Recording/Verifying MFCU IOCS (Object)	ID	ARGEN1	• Reads, punches, and prints cards from both hoppers • Provides error recovery
Line Printer—Dual Feed Carriage Printer IOCS (Object)	IN	ARGES1	• Initiates all valid printer operations for dual feed carriage control • Provides error recovery
Line Printer—Single Feed Carriage Printer IOCS (Compiler)	IO	ARGEW1 (wait call) ARGEW2 (execute call) ARGEW3 (overflow call)	• Initiates all valid printer operations for single feed carriage control • Provides error recovery
Line Printer—Single Feed Carriage Printer IOCS (Object)	IM	ARGEQ1	• Initiates all valid printer operations for single feed carriage control • Provides error recovery
Punch-Feed 1442 IOCS (Object)	IV	ARGEY1	• Punches cards from both hoppers • Provides error recovery
Punch Only/ Secondary Hopper MFCU IOCS (Compiler)	IJ	ARGEF1 (execute call) ARGEF2 (wait for cards to clear transport ARGEF3 (wait call)	• Punches cards from the secondary hopper • Provides error recovery
Read Column Binary 1442 IOCS (Object)	IS	ARGFA1	• Reads cards in column binary format • Provides error recovery

Descriptive Name	*Chart*	*Entry Point*	*Function*
Read Only/Both Hoppers MFCU IOCS (Object)	II	ARGEE1	• Reads cards from the primary or secondary hopper • Provides error recovery
Read Only/Primary Hopper MFCU IOCS (Compiler)	IE	ARGEI1 (execute call) ARGEI2 (wait call)	• Reads cards from the primary hopper • Provides error recovery
Read Only/Primary Hopper MFCU IOCS (Object)	IF	ARGEC1	• Reads cards from the primary hopper • Provides error recovery
Read Only/Secondary Hopper MFCU IOCS (Compiler)	IH	ARGEB1 (execute call) ARGEB2 (wait for cards to clear transport) ARGEB3 (wait call)	• Reads cards from the secondary hopper • Provides error recovery
Read Only 1442 IOCS (Compiler)	IU	ARGFC1 (execute call) ARGFC2 (dummy wait call) ARGFC3 (wait call)	• Reads cards from the hopper • Provides error recovery
Read Only 1442 IOCS (Object)	IT	ARGEX1	• Reads cards from the hopper • Provides error recovery
Read Primary/ Punch Secondary MFCU IOCS (Compiler)	IG	ARGEK1 (wait for cards to clear transport) ARGEK2 (punch wait call) ARGEK3 (punch execute call) ARGEK4 (read execute and wait call)	• Reads cards from the primary hopper • Punches cards from the secondary hopper • Provides error recovery
Read/Punch-No Feed 1442 IOCS (Object)	IR	ARGEZ1	• Reads cards from the hopper • Punches cards from the hopper • Provides error recovery

Flowcharts are identified in this publication in the following manner:

- A flowchart consisting of a single page is identified by a unique pair of letters.

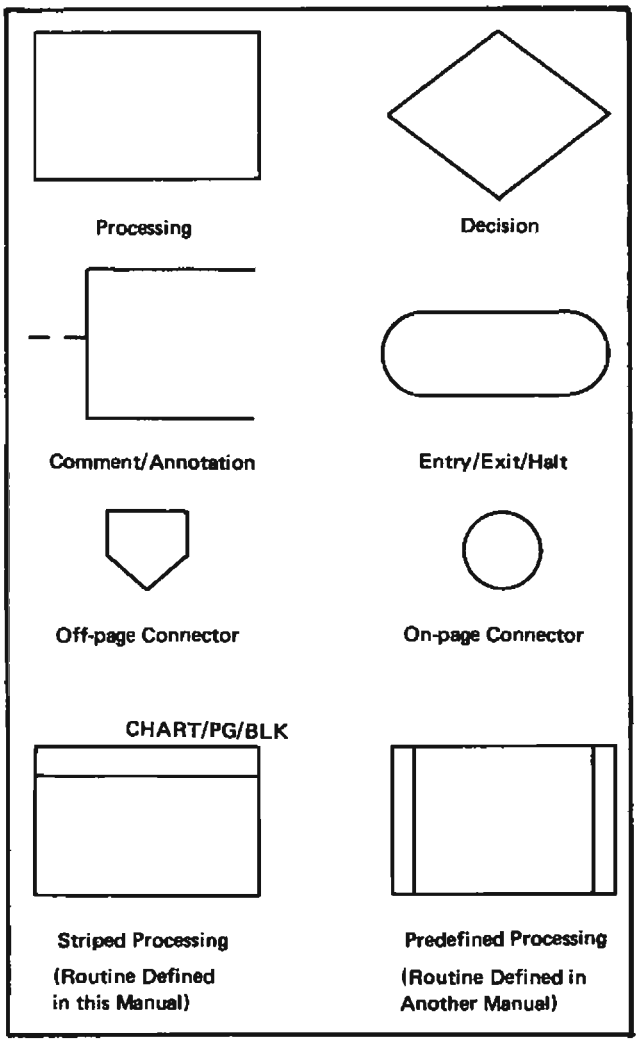
  Example: AA, AB, AC

- If a flowchart consists of multiple pages, each page is identified by the same pair of letters, but each page has a unique number.

  Example: First page, CA-01
  Second page, CA-02
  Third page, CA-03, etc.

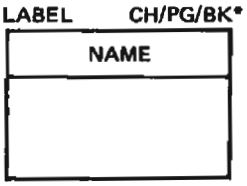
- Each part has been assigned two sets of flowchart identifying letters. Only after the first set has been completely used, i.e., AA-AZ, will the second set be used.

The flowchart symbols used in this PLM are:



1. The striped processing block indicates entry to a module or routine which is flowcharted and/or described in this logic manual.

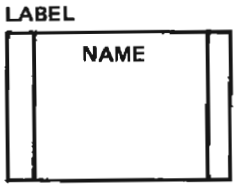
Example:



*CH/PG/BK indicates the flowchart, page, and block identification where the module or routine is flowcharted. If it is not flowcharted, a note gives the location of the description of that routine.

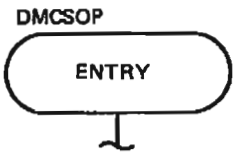
2. Predefined processing indicates a module or routine flowcharted in the *IBM System/3 Disk Systems Data Management and Input/Output Supervisor Logic Manual,* SY21-0512.

Example:



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

Example:

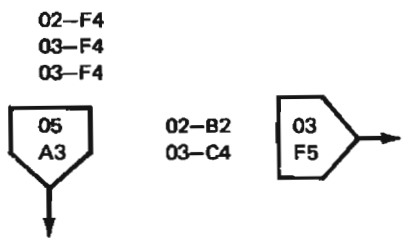


4. Off-page connectors are used to reference between different page of the same chart ID. Off-page connectors leaving a page contain the page number and block number of their destination.

Example:

Off-page connectors contain the page 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:



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

Absolute Card Loader

MFCU IOCS Routines

Printer IOCS Routines

IBM System/3 Printed in USA SY21-0521-1

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)