CONTROL DATA CORPORATION • COMPUTER EQUIPMENT GROUP
6000 Software Development DIVISION

DOCUMENT CLASS IMS CHANGE NO. 1 DATE 5/13/69 PAGE NO. 0
PRODUCT NAME EXPORT/IMPORT 200
PRODUCT MODEL NO. E034 MACHINE SERIES 6400/6500/6600
DEPT. NO. 6233 PROJECT NO. 3B701

SUBMITTED		REVIEWED			
PROJ. MGR.	DATE	DEPT. MGR.	DATE	DIRECTOR	DATE
				FINAL APPROVAL	

PRB CHAIRMAN DATE	☐ PPB DATE ☐ GEN MGR ☐ OTHER

INTERNAL MAINTENANCE SPECIFICATIONS

FOR

EXPORT/IMPORT 200

VERSION 1.0

© Copyright Control Data Corp. 1970

Contained herein are software products Copyrighted by Control Data Corporation. Reproduction of the copyright notice must appear on all complete or partial copies.

TABLE OF CONTENTS

I. EXPORT/IMPORT 200 SYSTEM

1.0 Introduction

1.1 Introduction and General Information

1.1.1 Routine Names

1.1.2 EXPORT PP Memory Allocation

1.2 Block Diagram of Components

1.3 General Description of a Processor

1.4 General Description of EXEC

1.5 General Description of IODC

2.0 Documentation of the System

2.1 Detailed Description of Calls to EXEC from a Processor

2.2 Detailed Description of EXEC Flags and Pointers

2.3 Details of IODC and EXEC Routines {1EU}

2.3.1 Narrative of IODC and EXEC Routines

2.3.2 Flow Charts of IODC and EXEC Routines

2.4 Details of the Reader Processor {4RU}

2.4.1 Narrative of the Reader Processor Routines

2.4.2 Flow Charts of the Reader Processor Routines

2.4.3 Reader Processor Save Area amd CM Buffer

TABLE OF CONTENTS (Continued)

2.5 Details of the Printer Processor (4PU)

2.5.1 Narrative of the Print Processor Routines

2.5.2 Handling of Non-Standard Print Files

2.5.3 Flow Charts of the Print Processor Routines

2.5.4 Print Processor Save Area and CM Buffer

2.6 Details of the Message Processor (4MU)

2.6.1 Narrative of the Message Processor Routines

2.6.2 Syntax Analysis Section of the Message Processor

2.6.3 Flow Charts of the Message Processor Routines

2.6.4 Message Processor Save Area and CM Buffer

2.7 Memory Allocation and EXPORT Initialization (EI2, ALC)

2.7.1 Narrative of EI2 and ALC

2.7.2 Flow Charts of EI2 and ALC Routines

2.8 Job Card Processing (1JU)

2.9 Direct Cell Assignments as Used by Each of the Overlays

2.9.1 IODC Direct Cell Allocation

2.9.2 EXEC Direct Cell Allocation

2.9.3 Processor Direct Cell Allocation

TABLE OF CONTENTS (Continued)

3.0 Installation Information

3.1 Installation of EST

3.2 Terminal Identification

3.3 Job Name Identification

3.4 Driver Installation Information

3.4.1 Introduction

3.4.2 Installation Options

3.4.3 Multiplexor Port Definition Macro

3.4.4 Subsystem Definition Macro

4.0 Other Information

4.1 Coding Conventions

4.2 Character Set

4.3 Central Memory Requirements

4.4 Bringing Up Central Site Software

4.5 EXPORT/IMPORT 200 File Definitions

TABLE OF CONTENTS (Continued)

II. DRIVER

Part I
EXTERNAL INTERFACE SPECIFICATION

1.0 Introduction

2.0 Central Memory Storage Map

3.0 System Pointers

4.0 Equipment Table

5.0 Multiplexor Status Table

6.0 Subsystem Tables

6.1 Subsystem Name and Status Table

6.2 Buffer Request Word

6.3 System Working Storage Area

7.0 Line Table

7.1 Line Status Word

7.2 Line Request Word

7.3 Line Storage Area

7.4 Line Message Area

8.0 {Section Deleted}

9.0 CM Buffers

10.0 Terminal Idling

11.0 Control Point Drop

11.1 Fatal Error Drop

11.2 Normal Operator Drop

TABLE OF CONTENTS (Continued)

II. DRIVER

Part II
INTERNAL SPECIFICATION

1.0 I/O Driver Philosophy

1.1 Data Handling

1.2 Buffer Handling

2.0 I/O Driver Initialization

3.0 Main Processing Loop Narrative

3.1 Main Flow Diagrams

4.0 Key to Task Flow Diagrams

5.0 Output Task Flow Narrative - 200 USER Terminal

5.1 Output Task Flow Diagrams - 200 USER Terminal

6.0 Input Task Flow Narrative - 200 USER Terminal

6.1 Input Task Flow Diagrams - 200 USER Terminal

7.0 Output Task Flow Narrative - Teletype Terminal

7.1 Output Task Flow Diagram - Teletype Terminal

8.0 Input Task Flow Narrative - Teletype Terminal

8.1 Input Task Flow Diagram - Teletype Terminal

TABLE OF CONTENTS (Continued)

II. DRIVER

Part II (Continued)

9.0 Miscellaneous Routines

10.0 Subsystem Zero - Transient PP's, 1SD and 1ID

10.1 Subsystem Zero Narrative

10.2 Flow Charts for 1SD and 1ID

11.0 Termination Routine Narrative (1PR)

11.1 Termination Routine Flow Chart (1PR)

12.0 Table Usage

12.1 Table Usage, One for Each Port

12.2 Other Tables, Not One per Port

13.0 Direct Cell Usage

1.0 INTRODUCTION

The EXPORT/IMPORT 200 system utilizes two peripheral processors. One is dedicated and handles I/O with the multiplexor. The other is transient and directs all operations associated with EXPORT/IMPORT after the remote user calls the system into action. The transient PP performs basic calls to the SCOPE system to transfer data between central memory and the disk, and it interprets and responds to user requests. A description of the transient PP and its associated central memory allocator follows.

1.1. General Information

The transient PP program is split into two levels. One, called the processor level, contains the basic flow of how card reading, line printing, and display functions are sequenced. There are three sections, all called processors: reader processor, printer processor, and message processor. The reader processor sequences through the operations necessary to read cards. It looks for commands associated with reading,

requests storage, displays reader associated messages like "READER NOT READY", performs trailing blank deletion, and makes reader CIO calls.

The printer processor performs similar functions: requests storage for printer data buffers, performs print compression and PM message detection, makes CIO calls, and handles operator communication which is associated with printer data ("PRINTER NOT READY" and ending a print job, for example). There are other functions which are handled by the message processor. These are the creation of the H and B displays, various message handling, and some command processing which is not directly related to the flow of card data or print data.

The other level of the transient PP program performs basic requests for the processors. It schedules the order in which processor requests are performed. Some of its functions are:

To get a block of card data from the reader,

To send a block of print data to the printer,

To set up buffer areas,

To send a set of data to the display,

To wait a short while and return control to a processor.

The processors make simpler calls to this level of control to get their job done. One processor is not aware that several requests from other processors may be completed before its request is answered. When the processor is given control again, the last request is normally completed so that the processor can go on again. This lower level that is called by the processor consists of two sections. The EXEC (executive) schedules which requests from the processors are performed next and decides what processor should receive the responses from the terminal. The EXEC calls upon a routine called IODC (Input Output Driver Caller) which sets up a single send and receive of data to and from the terminal. It is while the system is in the routine IODC that the PP is released and the transfer of data to and from the terminal is performed. When IOD (Input Output Driver) calls the EXPORT/IMPORT subsystem back into a PP, the control is normally given to IODC first, which then determines that the transfer has been made with no errors. IODC will return control to the EXEC, which decides what processor is returned to next. Control is finally returned to a processor to determine which function has to be performed next on this stream of data.

To allow communication between these various sections of the EXPORT/IMPORT system, various interfaces have been defined. The center of control is in the processor. Each processor operates independently and communicates to and from the executive through calls to the EXEC (return jumps to specific entry points of the EXEC) and by information stored in the line table entries.

This is the only information which is stored when the PP program of the EXPORT/IMPORT subsystem drops its PP. The information is processor dependent. The information associated with one processor is called the director for this processor. There are three processors, and therefore three directors. They are stored in the line table entry for each assigned line.

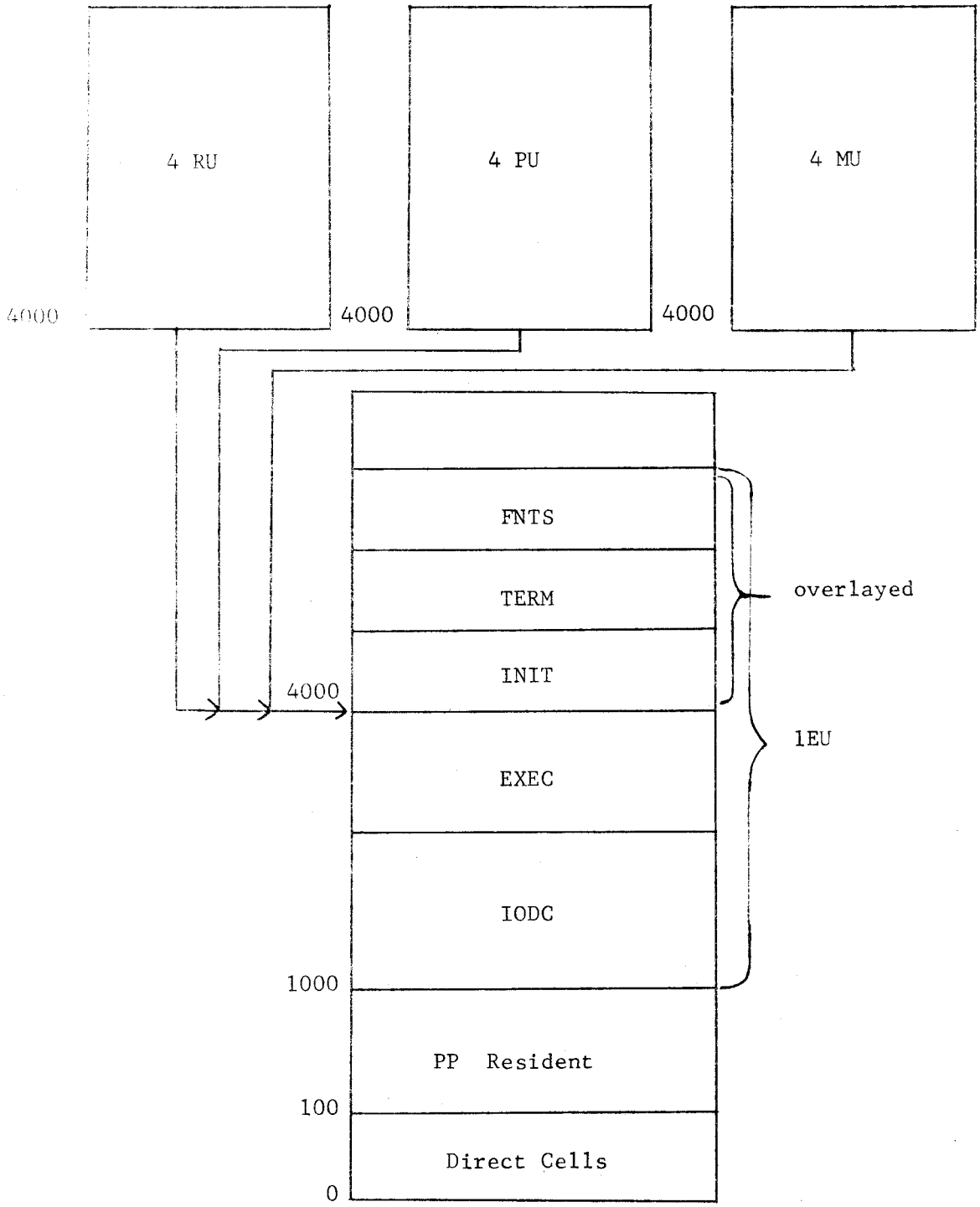
Since EXEC and IODC must also store some information about the status of the line, there is also one word of IODC and EXEC storage in each assigned line table entry.

1.1.1 Routine Names for EXPORT/IMPORT 200

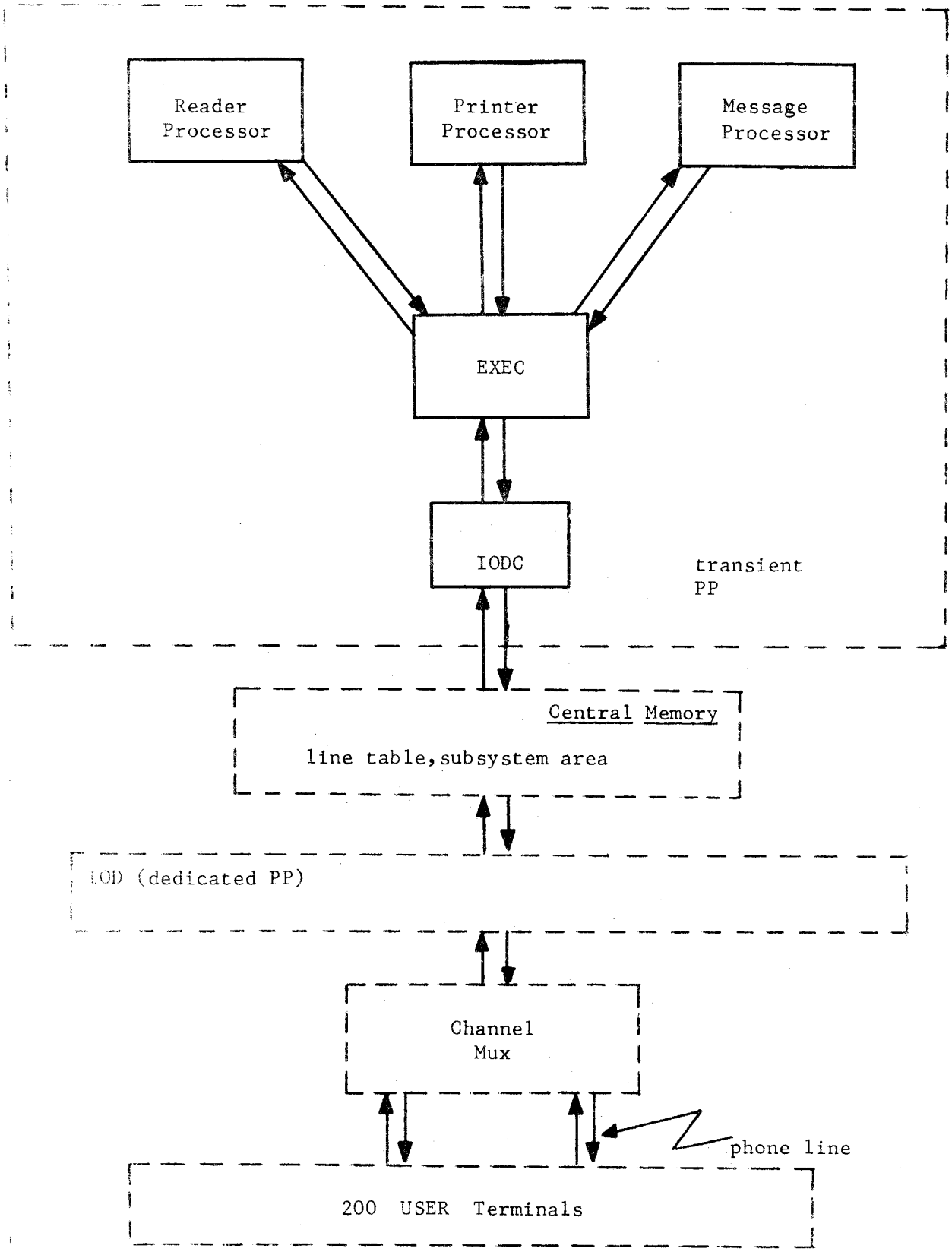
ROUTINE	DESCRIPTION
1EU	is the master routine for the transient PP. It contains EXEC and IODC. It schedules the operations of the processor overlays.
4MU	is the message processor overlay of the transient PP. It handles syntactical analysis.
4RU	is the reader processor overlay of commands. It handles the sequencing of card reading operations and makes CIO calls for card data.
4PU	is the printer processor overlay. It performs functions similar to 4RU but for print data.
ALC	is a central memory program which does memory allocation when RESPOND II is not running.
EI2 (deck name 1AU)	is an initialization routine which is called to bring the components of EXPORT/IMPORT 200 to a control point.
1JU	is a short routine which can be called into another transient processor to check the validity of job cards.
1OD (deck name NOD)	is the initialization overlay for the peripheral processor which is dedicated to communications. It picks up the EST entry and sets up information for 1OD before calling it into the same PP.
1OD	is the main routine of the dedicated communications processor. It is the set of routines which perform the input output with the multiplexor.
5OD	is a set of tables which are called into the dedicated processor to overlay some of the initialization code in 1OD.

ROUTINE	DESCRIPTION
1PR	is a routine used to terminate the operations of the dedicated communication processor. It restores the PP resident of 1OD.
1ID	is a transient PP for subsystem zero. It assigns subsystems, sets up TERMINAL IDLE M PP and SUBSYSTEM NOT AVAILABLE messages, and does cleanup before dropping.
1SD	is a transient PP program associated with 1OD. 1SD is in rapid monitor recall and performs the services required by the subsystems; namely, loading, requesting storage, assigning central display, etc. In addition, 1SD produces the system status displayed at the control point.

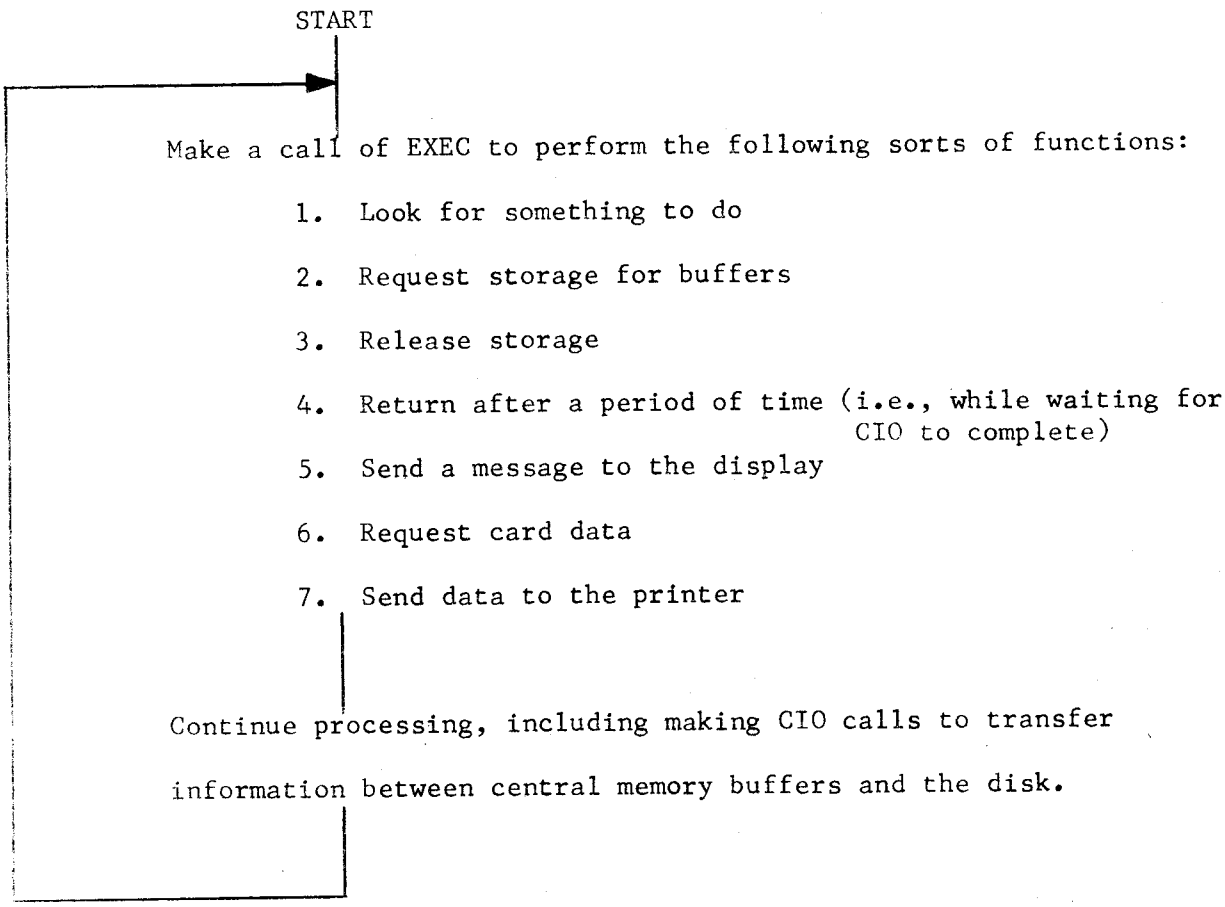
1.1.2 EXPORT PP Memory Allocation



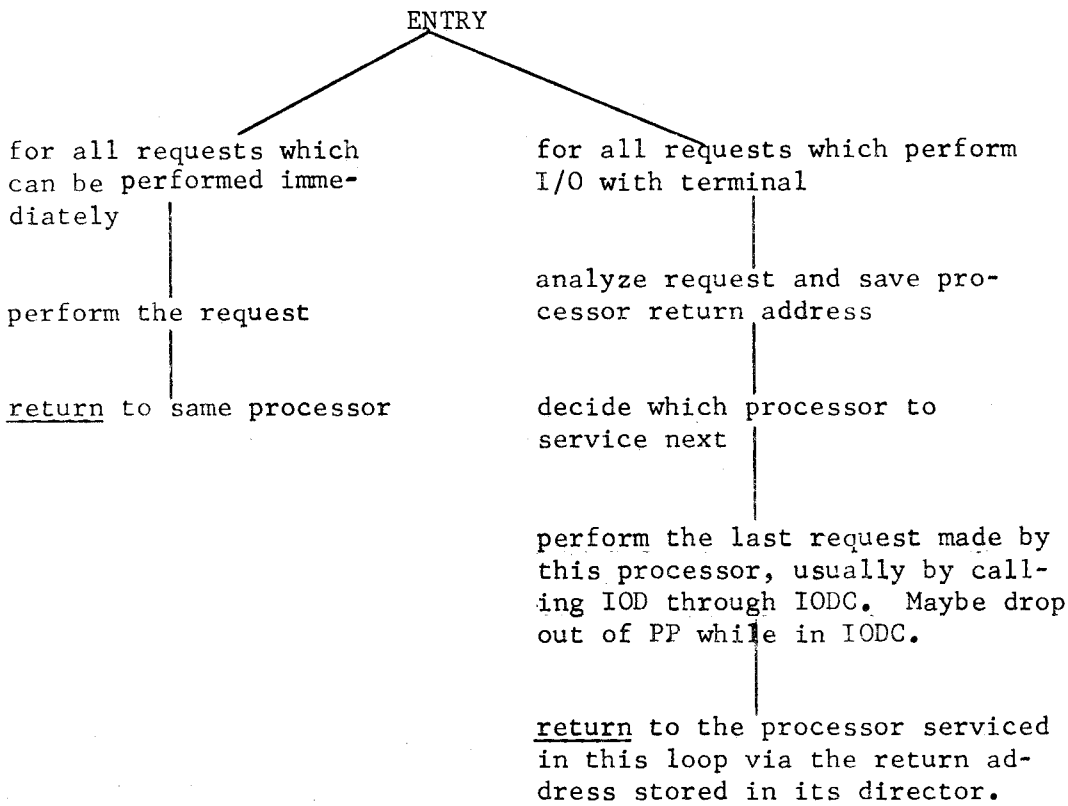
1.2 Block Diagram of EXPORT/IMPORT Components and Communication Lines



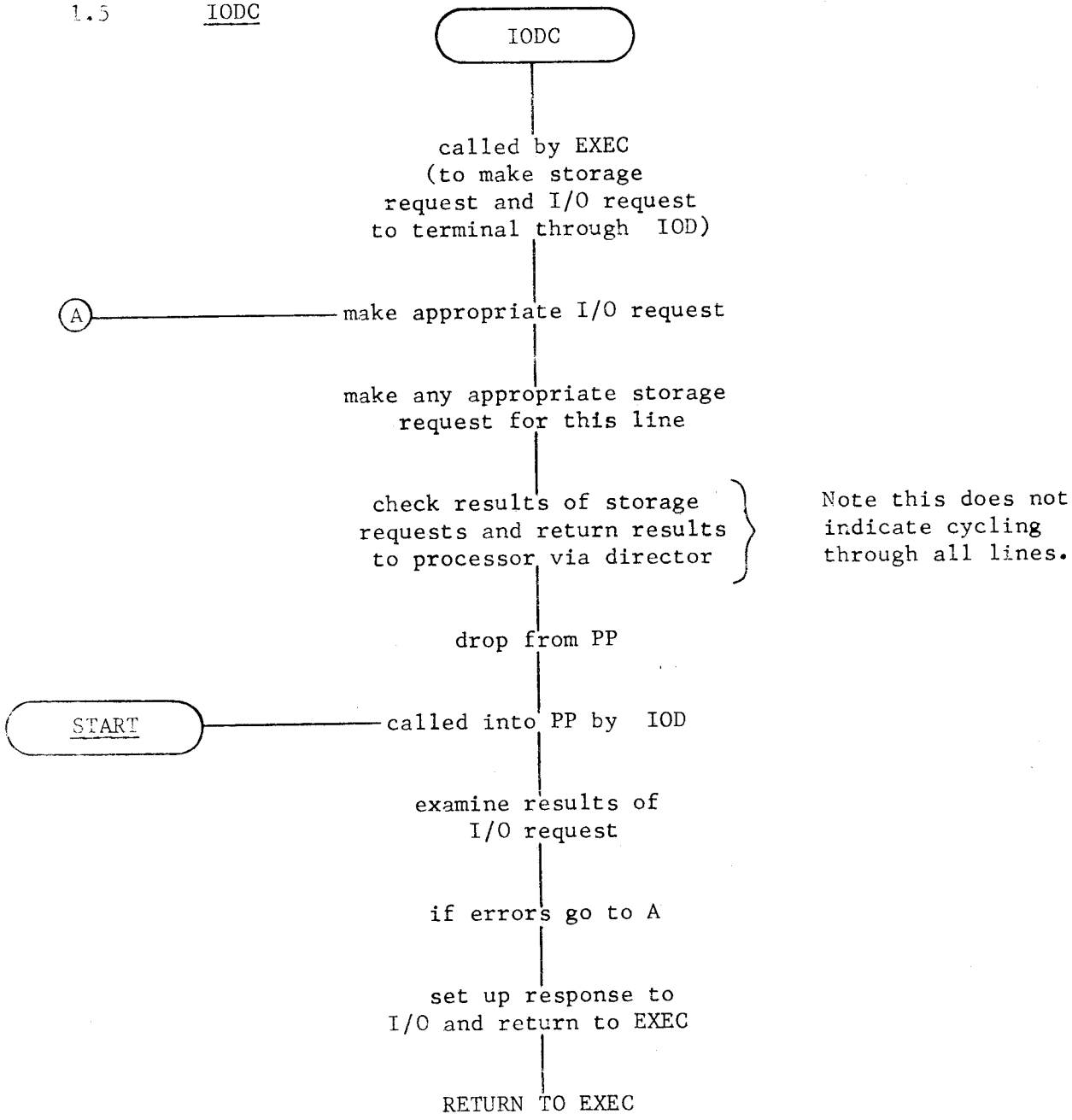
1.3 General Description of a Processor



1.4 EXEC


ENTRY
for all requests which can be performed immediately
perform the request
return to same processor
for all requests which perform I/O with terminal
analyze request and save processor return address
decide which processor to service next
perform the last request made by this processor, usually by calling IOD through IODC. Maybe drop out of PP while in IODC.
return to the processor serviced in this loop via the return address stored in its director.

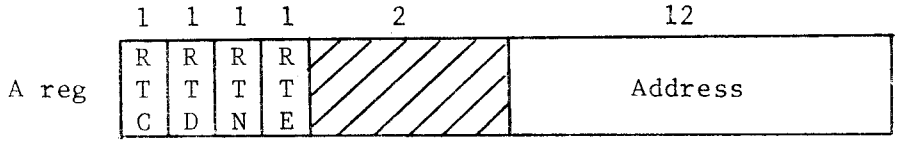
1.5 IODC


IODC
called by EXEC (to make storage request and I/O request to terminal through IOD)
A
make appropriate I/O request
make any appropriate storage request for this line
check results of storage requests and return results to processor via director
Note this does not indicate cycling through all lines.
drop from PP
START
called into PP by IOD
examine results of I/O request
if errors go to A
set up response to I/O and return to EXEC
RETURN TO EXEC

2.0 DETAILED DOCUMENTATION

2.1 Calls to EXEC

The calls that can be made of the EXEC by the processors show how some of the communication is performed. A description of the calls follows. When control is returned to a processor the A register will contain some bits which describe what the result of the call has been.



RTC if set, a command has been received

RTD if set, the requested operation has been done

RTN if set, the remote device (printer or reader) associated with this call is not ready

RTE if set, an error in a card code was detected

Address, used to return an address after certain calls

IDLE

calling sequence: RJM IDLE

function: IDLE is called by a processor when it has no further processing to do and is therefore idle. If all three processors are idle, EXEC will display the message,

* HH.MM.SS MM/DD/YY XY *

and poll the terminal. Any storage which was assigned when IDLE is called will be automatically released unless A≠0. In this case, storage is not released and, if called by the print processor, print files will not be picked up for printing.

returns on: Control is returned to a processor when a command is available for the processor and/or there is some function for it to perform. For the read processor, only a command (R;E,CR; LOGOUT) will cause a return. For the print processor, a command (E,LP;A;A,n; O,jj; O,jj,xx; S;C; LOGOUT) and/or a print file available for printing will cause a return. In the latter case, the low order 12 bits of the A register will contain the absolute address of

the FNT entry. For the message processor, any command (as yet unanalyzed) and/or an operator message in the control point area will cause a return.

	Processor	RTC	RTD	RTN	RTE	Address
Command	READ	1	0	0	0	0
Command	PRINT	1	0	0	0	0
Function + Command	PRINT	1	1	0	0	FNT address
Function	PRINT	0	1	0	0	FNT address
Command	MESSAGE	1	0	0	0	0
Function + Command	MESSAGE	1	1	0	0	0
Function	MESSAGE	0	1	0	0	0

Note if PRINT call had A≠0, only a command return is possible.

READ

calling sequence: RJM READ

The relative address of the buffer to be used by IOD is in the BAD area of the director. BAD is set by a storage request.

function: A block of card data is requested to be sent from the terminal and to be put in the buffer assigned to this processor. The number of words read into the buffer will be returned in the BWC word of the direct cells.

returns on:

	RTC	RTD	RTN	RTE
done, no error, ready	0	1	0	0
done, no error, not ready	0	1	1	0
done, bad card code, ready	0	1	0	1
done, bad card code, not ready	0	1	1	1
not done, command	1	0	0	0

PRNT

calling sequence: RJM PRNT

The relative address of the buffer which contains the data is in the BAD field of the director. IOD will expect pointers to the buffer (STOP, RESET) to be updated by the processor.

function: To send print data to the terminal from a central memory buffer. Transmission errors are automatically recovered.

returns on:

	RTC	RTD	RTN	RTE
print done, no command	0	1	0	0
print done, command	1	1	0	0
print not done, command	1	0	0	0
print not done, not ready	0	0	1	0

MESG

calling sequence: RJM MESG

The A register must contain a PP address of a message to be sent to the display. the message is terminated by an end-of-message byte (0001_8) and must be less than 30 characters including the end of message. If the address is zero, a CM buffer whose address is in BAD will be used instead.

function: A message will be displayed for a short period of time and then may be wiped out by another function, such as reading or printing. The MESG request is used by a processor to display messages, such as PRINTER NOT READY. The message will be regenerated automatically if the display is clobbered by a read or print operation. If multiple processors call message, only one message will be displayed at a time.

returns on: A return is made to a processor when a command is received. A GΔ command will be sent only to the processor which is currently displaying a message.

STAL

calling sequence: RJM STAL

The amount of storage that will be assigned to a processor is obtained from the BBC parameter of the director. This field is set up at initialization time.

The BAF (buffer allocate flag) is set by EXEC and BAD (buffer address) is set by IODC when storage is allocated. The A register must be set as for MESG to display a message until storage is allocated.

function: To get a block of storage for a data buffer. The message will be handled in a fashion similar to the message of the MESG request.

returns on: The done bit is not used. If BAD is zero, storage was not assigned; if non-zero, BAD is the address of the beginning of a buffer (12 bit relative to RA *100). A command may also cause a return with BAD zero or non-zero. If the storage is desired and BAD=0, the call should be re-issued.

WAIT

calling sequence: RJM WAIT

function: The purpose of the WAIT command is to wait for a fraction of a second so that control can be given to other lines and processors, while something that the processor is waiting for is completed (normally a CIO call). No terminal communication is initiated by the request.

returns on: The processor gains control after a period of time or after a command is returned as a result of communication initiated by another processor.

CMRL

calling sequence: RJM CMRL

command release

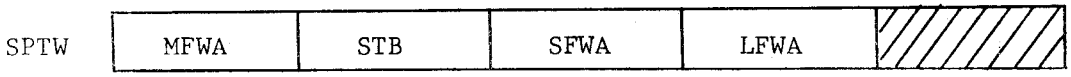
function: The EXEC needs to know when a processor has finished accessing a command in the message buffer so that further communication with the terminal can proceed. If EXEC went ahead, the memory buffer could be clobbered by a new command.

returns on: CMRL will return immediately and therefore the PP will not be over-written during the call. Direct cells 51_8-55_8 are changed by this call.

2.2 Detailed Description of EXEC Flags and Pointers

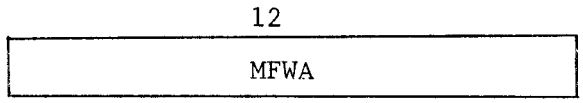
Note: In the following documentation of EXPORT's CM usage, only those areas used by EXPORT are indicated. Any area indicated by is neither used nor altered by EXPORT, but may be used by other programs. Any area indicated by is not used by EXPORT, but is set to zero by EXPORT.

I. System Entries



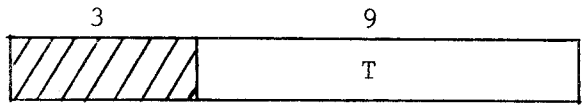
A. SPTW = System Pointer Word. SPTW is set by IOD when the system is initialized. 1EU does not alter SPTW. The address of SPTW (relative to RA) is communicated to 1EU via the low order byte of its PP input register.

1) MFWA = Multiplexor Status Table First Word Address.



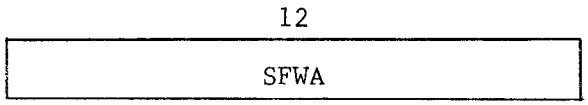
MFWA is used by 1EU to access the first word of the multiplexor status table which contains the system clock (see I.B).

2) STB = Subsystem and Terminal Count Byte



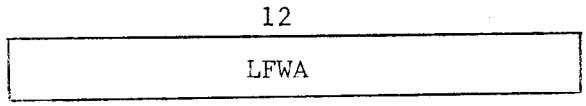
T = Terminal count. T is the number of line table entries in the Line Table. T is used by INIT when the line table is scanned to assign new lines to EXPORT.

3) SFWA = Subsystem Table First Word Address.

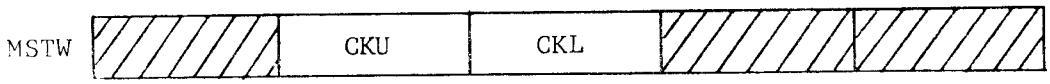


SFWA is used by INIT to compute the address of the EXPORT subsystem entry in the subsystem table.

4) LFWA = Line Table First Word Address.



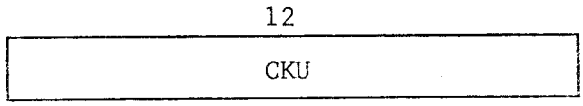
LFWA is used by INIT to establish the first line table entry when new lines are to be assigned to EXPORT.



B. MSTW = Multiplexor Status Table First Word.

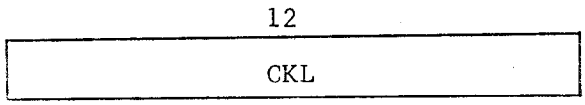
MSTW is set and altered by IOD. 1EU does not alter MSTW.

1) CKU = Clock Upper.



CKU contains the time, in milliseconds/4096, since the system was initialized. Thus, the low order bit of CKU changes every 4.096 seconds.

2) CKL = Clock Lower.



CKL contains the time, in milliseconds (modulo 4096), since the system was initialized. CKU and CKL are used by 1EU for timing purposes.

II. Subsystem Table Entry

SNSW	0					SSB
SSRW	1	SRB	ABC	AAD	RBC	RAD
ST1W	2	IPT	ADB	ALA	FST	CDA
ST2W	3	DPF	RTC	SRF		
	4					
	5	Not Used				
	6					
	7					

A. SNSW = Subsystem Name and Status Word.

SNSW is the first word of the subsystem table entry. Only SSB is used by 1EU and is altered by both IOD and 1EU. The address of SNSW is SFWA+8* NEXP where NEXP=2 is the number assigned to the EXPORT subsystem.

1) SSB = Subsystem Status Byte.

4	1	1	2	2	1	1
RK	SFT	KL		CD	ST	SL

RK = Recall clocK. RK is set by 1EU to the time it wishes IOD to reload the subsystem. RK is set to the value of the high order 4 bits of CKL plus some constant indicating the recall period. 1EU requests recall periods of 3-3/4 sec. when no lines are assigned, 1 sec. when lines are assigned, and 1/4 sec. when storage is being allocated. RK is only used by IOD when KL=1 (see below).

SFT = Subsystem First Time flag. SFT is set by IOD when a line has requested assignment to the EXPORT subsystem. When set, INIT will scan the entire line table and enter those lines which have LFT=1*(and are otherwise assignable) into the string of assigned lines. INIT resets SFT to zero.

KL = clocK fLag. KL is set by IODC just before dropping its PP to indicate to IOD that it desires a timed recall according to the value in RK.

CD = Central Display flag. CD is used to allocate the central display to a particular subsystem. When a user enters an M command, IODC will set CD=1. When IOD assigns the central display to EXPORT, it sets CD=2. After the command is written to the central display, IODC sets CD=0 releasing the display for reassignment. (1EU uses CDA, CAF, and CAL described below, to sub-assign the console to the individual lines.)

*See Section 3A

ST = Subsystem sTatus. ST indicates whether or not a subsystem is loaded. ST is set to 1 by IOD when it loads 1EU and 1EU resets ST to zero when it drops.

SL = Subsystem Lock. SL is used to lock writing of SSB onto either IOD or 1EU. When IOD loads 1EU, it sets ST=1 and SL=0, thus allowing 1EU to alter the word. When 1EU drops, it inverts both flags setting ST=0 and SL=1, allowing IOD to alter the word. When the operator drop flag is set and all the lines assigned to EXPORT are released, TERM sets ST=SL=0, indicating that the subsystem is no longer available.

B. SSRW = Subsystem Storage Request Word.

SRB	ABC	AAD	RBC	RAD

SSRW is used by 1EU, IOD and a CM resident storage allocation program. IOD does not alter this word, but checks it periodically and recalls the CP if a request is being made. IODC and the allocator alter the word according to a lock bit in SRB.

1) SRB = Storage Request Byte.

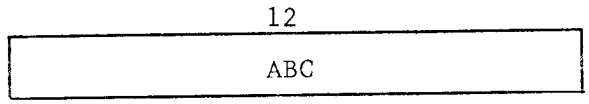
SRB is set by IODC and reset by the allocator.

1	11
RQ	0

RQ = ReQuest flag. Normally, RQ=0 indicating IODC may alter SSRW. When IODC wishes to allocate or release storage, it sets RQ=1. When IOD sees RQ=1, it starts the allocator which attempts to execute the request and then sets RQ=0.

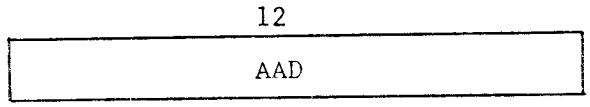
2) ABC = Allocate Block Count.

ABC is set by IODC and not altered by the allocator.



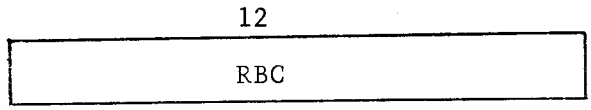
ABC is used to specify the number of 100_8 CM word blocks to be allocated. If ABC=0, no allocation is attempted.

3) AAD = Allocate ADdress. AAD is cleared by IODC and set by the allocator.



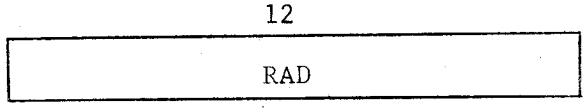
AAD specifies the first word address/100_8 of CM allocated to EXPORT. The allocator sets this address when RQ=1 and storage can be allocated. If storage is not available, AAD is not set. If AAD≠0 or ABC=0, no allocation is attempted. AAD is relative to RA.

4) RBC - Release Block Count. RBC is set by IODC and not altered by the allocator.



RBC is used to specify the number of 100_8 CM word blocks to be released. If RBC=0, no release is performed.

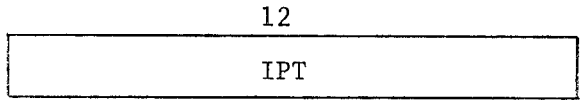
5) RAD = Release ADdress. RAD is set by IODC and cleared by the allocator.



RAD specifies the first word address/100_8 of CM to be released by the allocator. The allocator will always perform release requests and then reset RAD to zero (when RQ=1). If RAD=0 or RBC=0, no release is performed.

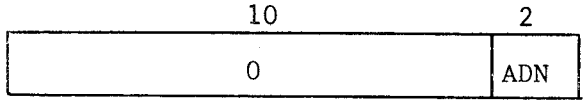
C. ST1W = Subsystem Temporary 1 Word. ST1W is used only by 1EU for temporary storage purposes.

1) IPT = Initial PoinTer.



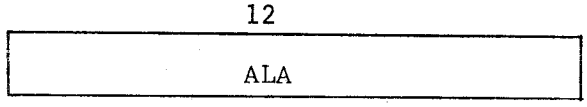
IPT is the address (relative to RA) of the first line table entry assigned to EXPORT. If IPT=0, no lines are assigned.

2) ADB = Allocate Director Byte.



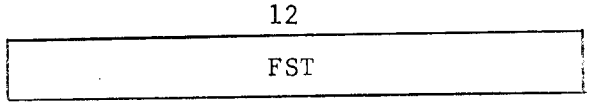
ADN = Allocate Director Number. ADN and ALA (see below) are used by IODC to specify the line and processor currently attempting storage allocation. IODC sets ADN and ALA when the request is made and continually attempts allocation until either the storage is allocated or the request is aborted by the processor. When ADN=ALA=0, no request is in progress.

3) ALA = Allocate Line Address.



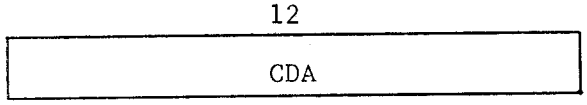
ALA is used by IODC to specify the address (relative to RA) of the line table entry currently attempting storage allocation. If ALA=0, no request is in progress.

4) FST = FNT Scan Time.



FST is the time (from CKU and CKL), in fourths of seconds modulo 100_8, of the last FNT scan for output files. When 1EU is loaded, FNTS checks FST against the current time. If the current time is 1 second or more greater than FST, FST is reset and the FNT is scanned.

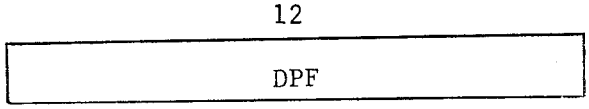
5) CDA = Central Display Address.



CDA is used by IODC to specify the address (relative to RA) of the line table entry currently requesting (CD=1) or currently assigned to (CD=2) the central display. CDA remains set until the requesting line aborts the request. When CDA=0, CD=0 and no line is assigned to the central display.

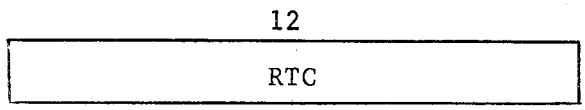
D. ST2W = Subsystem Temporary 2 Word. ST2W is used only by 1EU for temporary storage purposes.

1) DPF = DroP Flag.



DPF is used by 1EU to indicate when the operator drop flag has been set and when a line is to be dropped from assignment to EXPORT. Bit 2 (value 4) is set by TERM when the operator drop flag is found and is not reset. Bit 0 (value 1) is set by IODC when EXEC calls IODC to flush a line. This is reset by TERM when the line is flushed. Both requests are performed by TERM when 1EU is loaded. For a line flush, the subsystem ID is reset and the string pointers re-linked to remove the line from the string. For an operator drop the PP is dropped after all lines have been flushed (IPT=0).

2) RTC = ReTurn Code.



RTC is the last return code sent to EXEC by IODC. This is not required between PP recalls, but is saved in CM for debugging purposes. The values are specified as (bad card codes) $*10_8$ + (data received) *4 + (read type i.e., E1, E2, or E3). The normally returned values are:

0 = no read (i.e., rejects after a write, poll or poll only call, or write completed after a write only call)

1 = RE1, no data received

2 = RE2, no data received

3 = RE3, no data received

5 = RE1, data received

6 = RE2, data received

7 = RE3, data received

16 = RE2, data received, bad card codes

17 = RE3, data received, bad card codes

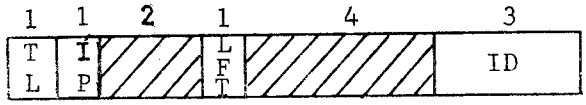
3) SRF = Storage Request Flag. SRF is a count which is set to 2 when storage has been assigned by the allocator. When SRF is non-zero IODC will decrement SRF and re-scan the export string without dropping the PP. SRF is used to speed up storage allocation to the processors.

III. Line Table Entry

CLSW	0	LSB	TSB	RSB	PNB	
CLRW	1	LRB	CBB		RBB	WBB
CIES	2	ESB	ISB	ICB	IRT	NPT
	3	Processor 1 Storage				
	4					
	5	Processor 2 Storage				
	6					
	7	Processor 3 Storage				
	8					
	9	Message Buffer				
	10					
	11					
14B	W. CPTID	Account number				
		TID	UI			
16B	W. CPCCT	C.CRCT = 0 Cards Read	C.LPCT = 2 Lines printed	C.ERTP=4	R1	R1 X command entered
18B	W. CPJBC	Jobname From last job card				
20B	W. FST					
21B						

A. CLSW = Current Line Status Word. CLSW is used to indicate I/O status upon completion of transmissions. TSB, RSB, and PNB are are set by IOD and never altered by EXPORT, while LSB is altered by both IOD and IODC.

1) LSB = Line Status Byte. LSB is altered by IOD when TL=0 and by IODC when TL=1.



TL = Terminal Lock. TL is set to 1 by IOD when I/O communication with the line has stopped. This may be caused by:

a) An error detected on a read or write

b) A write acknowledgment after a write only request (SD≠0, RD=0, see II. A)

c) A read after a write-poll or a poll only request (SD, RD≠0, or SD=0, RD≠0)

d) No request being made (SD=RD=0)

TL is set to zero by IODC to allow IOD to check the I/O request and perform the operation (if any). When TL=0, the subsystem ignores CLSW entirely.

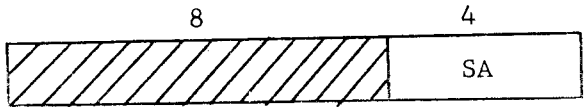
IP = In Process flag. When TL=1 and IP=1, a transmission error has occurred and IODC will request a retry by clearing TL.

LFT = Line First Time Flag. LFT is set to 1 by IOD when the line goes from the "Terminal Idle" state to the state where it is assigned to EXPORT. LFT is set only once (when ID is set) and is reset by INIT after the line is entered into the string of assigned lines.

ID = IDentification Code. ID identifies the subsystem assigned to this line. Once set by IOD, ID will not be changed until TERM sets it to zero when the line is flushed.

2) TSB = Terminal Status Byte. TSB is set by IOD and not altered by EXPORT.

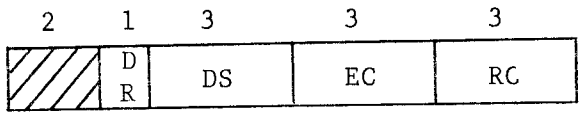
8	4
	SA



SA = Site Address. SA is the terminal site address and is used by the subroutine CJCB in 1EU to select the second character of the terminal ID, namely U-9 for SA=0-17_8 respectively. SA is set by IOD and is not changed until ID is set to zero.

3) RSB = Reply Status Byte. RSB is set by IOD every time TL is set to 1 and is not altered by EXPORT.

2	1	3	3	3
	DR	DS	EC	RC



DR = Data Received Flag. DR is set to 1 if data (other than an E code) has been received.

DS = Device Status. DS indicates the type of E code received after a read.

0 or 1 = RE1

2 = RE2

3 = RE3

EC = Error Code. EC is set whenever any kind of error condition is detected. The codes are:

0 = No error

1 = Data lost due to insufficient buffer space (ignored by IODC)

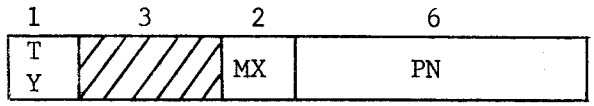
2 = Bad card codes

3-6 = Parity errors, hardware errors, etc. (IODC requests retransmission for these errors)

7 = Data phone time out (IODC requests retransmission until 37_8 consecutive timeouts occur at which point the line is logged out; i.e., someone has hung up the phone)

RC = Reply Code. RC indicates the reply received from the last write operation. IODC only checks this after polling to see if a read (RC=1) has occurred.

4) PNB = Port Number Byte. PNB is set by IOD when the system is initialized and is not altered by EXPORT.



TY = TeletYpe Flag. TY is set to 1 if the terminal is a teletype terminal. In this case, INIT will ignore the line and set ID=0 if LFT is set.

MX,PN = MultipleXor and Port Number. These indicate the actual hardware connection of the line and are used by the subroutine CJCB in 1EU to generate the 1st character of the terminal ID via an installation dependent table.

B. CLRW = Current Line Request Word. CLRW is set by EXPORT when TL=1. LRB, RBB, and WBB are used to specify an I/O transfer to be attempted by IOD after TL is set to zero. In certain cases, LRB is altered by IODC when TL=0. JCB and TOB are used by EXPORT only.

1) LRB = Line Request Byte. LRB is set by IODC and altered by IOD.

2	1	2	1	3	3
SD	0	RD	0	MT	DT

SD = Send Data Flag. When SD≠0, IOD is requested to write to the terminal from a buffer. The values are:

2 = CM Buffer

3 = Message Buffer

When the write is completed, IOD will set SD=0 and proceed to look at RD.

RD = Read Data Flag. When RD≠0, IOD is requested to poll repeatedly until a read is returned or until the subsystem sets RD=0. Data is read into a buffer according to the value of RD. The values are:

2 = CM Buffer

3 = Message Buffer

RD is not altered by IOD.

MT = Message Type. MT is always set to 5 (clear write) by IODC.

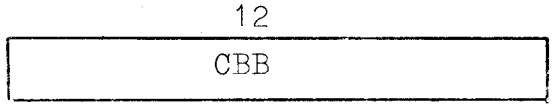
DT = Device Type. DT specifies the device to receive a write. The values are:

1 = Display

2 = Printer

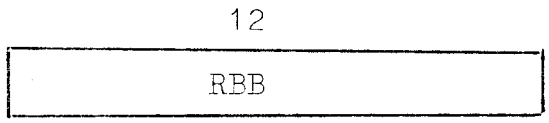
3 = Card Reader

2) CBB = Current Buffer Byte. CBB is zeroed by IODC and set by IOD.



If RD=2, CBB is set to the number of words written into the CM buffer. CBB is stored in BCW for the reader processor.

3) RBB = Read Buffer Byte. RBB is set by IODC and used by IOD.

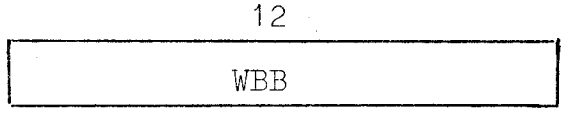


If RD=0, RBB is ignored.

If RD=2, RBB is the FWA/100_8 of a CM buffer.

If RD=3, RBB is the address (relative to RA) of the Message Buffer in the line table. That is the address of CLSW+9.

4) WBB = Write Buffer Byte. WBB is set by IODC and used by IOD.

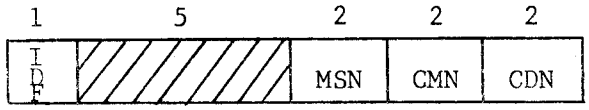


If SD=0, WBB is ignored

SD=2 or SD=3, interpretation is the same as for RBB.

C. CIES = Current IODC and EXEC Storage. CIES is totally internal to EXPORT. ESB and ISB are shared between IODC and EXEC, while ICB, IRT, and NPT are used only by IODC.

1) ESB = EXEC Status Byte. ESB is set only by EXEC and is used by IODC.



IDF = IDle Flag. IDF is set to 1 by EXEC after the idle message is displayed and is reset to zero when any processor leaves the idle state. It is used to prevent rewriting the idle message after each cycle through the processors.

MSN = MeSsage Number. MSN is set to the number of the processor which is currently displaying a message via a call to MESG or STAL. MSN is reset to zero when a G command (or a command for this processor) is entered. MSN is also cleared if storage is allocated during an STAL call.

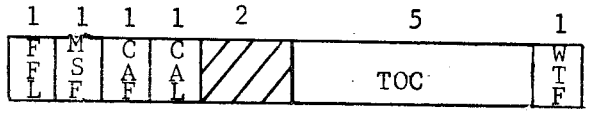
CMN = CoMmand Number. CMN is set by the message processor to the number of the processor which is to receive an entered command. When CMN is non-zero, normal processor cycling ceases until the "locked on" processor calls CMRL, which zeroes CMN. During the "locked on" period, all EXEC calls are executed as WAIT calls. (Commands which are processed

by the message processor are done so by an indirect call to IODC that bypasses EXEC's processor cycling.)

CDN = Current Director Number. CDN is the number of the processor that is currently calling IODC (via EXEC) or is currently executing. CDN=0 when all processors have called IDLE, or IDLE and WAIT.

2) ISB = IODC Status Byte. ISB is altered by both IODC and EXEC.

1	1	1	1	2	5	1
FFL	MSF	CAF	CAL		TOC	WTF



FFL = Flush FLag. FFL is set under the following conditions:

a. By IODC when 37_8 consecutive timeout errors have occurred

b. By TERM when the operator drop flag is set

c. By the message processor when a LOGOUT command is entered.

When FFL is set, IODC treats all write calls as nops and all write-poll or poll calls as a nop with an E1 read of the command LOGOUT. When all processors are idle and FFL is set, EXEC will wait until all storage is released by IODC and will then call IODC to flush the line. FFL is never reset until the line is reassigned by IOD.

MSF = Message Flag. MSF is set to 1 by 4MU when a message to all terminals has been sent to this terminal. MSF is cleared by FNTS when all the MSF flags are set. MSF is used to prevent the message from being cleared by FNTS until it has been displayed on all the terminals.

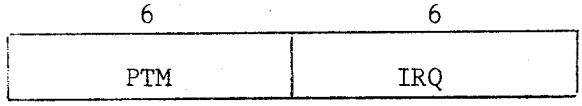
CAF Console Allocate Flag. CAF is set to 1 by a line's message processor when an M command is entered and then reset to zero after the message has been sent to the central site. When CAF is set, IODC will attempt to allocate the central site display for this line (via IOD). When the display has been allocated, CAL (see below) will be set to 1. CAF is not altered by IODC.

CAL Console ALlocated Flag. CAL is set to 1 by IODC when the central display is allocated to this line. CAL is reset to zero by IODC after CAF has been cleared by the line's message processor. CAL is not altered by the message processor.

TOC = Time Out Count. TOC is a count of consecutive time out errors. Whenever a reply is received that is not a time out error, TOC is reset to zero. When TOC reaches 37_8, FFL is set to logout the line. TOC is used only by IODC.

WTF = WriTe Flag. WTF is set to 1 by IODC after a write has been started and is reset to zero after a read has been completed. WTF is used by EXEC to prevent re-writing messages without an intervening read. WTF is zeroed by EXEC in certain cases when it is necessary to have the display over-written.

3) ICB = IODC Call Byte. ICB is set by IODC and is not used by EXEC.



PTM = Poll TiMe. PTM is set to the current value of IOD's clock (in ¼ seconds) at the beginning of a call to poll only. This is compared with the current time, on subsequent recalls of IODC, to allow IODC to stop polling every two seconds.

IRQ = IODC call ReQuest. IRQ is set each time IODC is called by EXEC. IRQ's value is the low order 6 bits of the A register at the time of the call. IRQ is interpreted as three 2-bit groups which describe the operation to be performed by IOD. The call is specified as (SD) $*20_8$ + (RD) $*14_8$ + DT. If SD=1, SD is changed to 3 and WBB is set to zero. Otherwise, WBB is set as specified by SD. RBB is set as specified by RD. The currently used call requests are:

00 = Nop

55_8 = WE1 from a Circular Buffer plus Poll

56_8 = WE2 from a Circular Buffer plus Poll

33_8 = WE3, No write Buffer, Poll into a Circular Buffer

75_8 = WE1 from the Message Buffer plus Poll

61_8 = WE1 from the Message Buffer

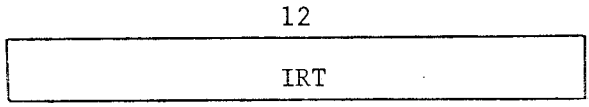
41_8 = WE1 from a Circular Buffer

15_8 = Poll and return after 2 seconds if no read

77_8 = Flush (no IOD action)

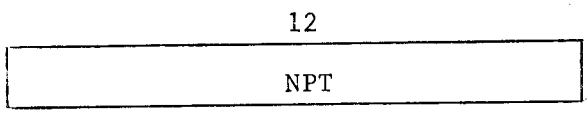
If RD=3, and a RE1 occurs, the three words in the Circular Buffer from RESET to RESET+2 (treated circularly) are copied into the Message Buffer by IODC. CURRENT is reset to RESET by IODC.

4) IRT = IODC ReTurn Address. IRT is set by IODC and is not used by EXEC.



IRT is the return address for the last call to IODC.

5) NPT = Next PoinTer. NPT is set by the initialization (INIT) and termination (TERM) sections of IODC and is used by IODC.



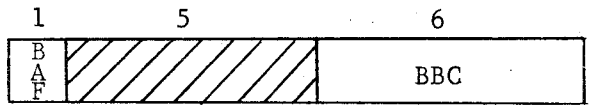
NPT is the address (relative to RA) of the next line currently assigned to EXPORT. If NPT=0, this is the last line in the string.

D. Processor Storage (identical for the three processors)

CLDC	DSB	BAD	DCB	ERT	MAD
CLPC	SAV1	SAV2	SAV3	SAV4	SAV5

1) CLDC = Current Line Director for the Current processor. CLDC is used by both IODC and EXEC to retain the status of a particular processor. CLDC is the name of the first word (the Director) of the processor storage area of the currently active processor (i.e., the processor indicated by CDN in CIES, see III.C.1). BAD is altered only by IODC, while DSB, DCB, ERT, and MAD are altered only by EXEC. DCB and MAD are also altered by FNTS in certain cases.

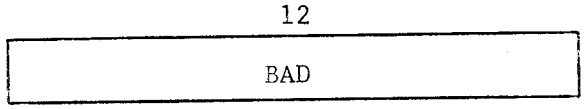
a) DSB = Director Storage Byte. DSB is set by INIT when the line is assigned and is altered by EXEC. IODC does not alter DSB.



BAF = Buffer Allocate Flag. BAF is set to 1 by EXEC when the processor calls STAL to allocate storage. BAF is reset to zero by EXEC when the processor calls IDLE with A=0. When BAF=1, IODC will attempt to allocate storage and indicate when it is allocated by setting BAD (see below). When BAF=0, IODC will release the storage and indicate the release by clearing BAD.

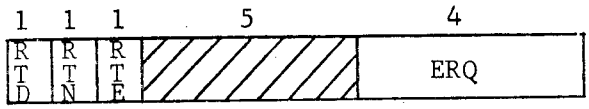
BBC = Buffer Block Count. BBC is set by INIT when the line is initialized and is not altered. BBC is the number of contiguous 100_8 word blocks of data to be assigned to the processor when BAF=1.

b) BAD = Buffer ADdress. BAD is set and reset by IODC and is not altered by EXEC.



BAD is set by IODC when BAF=1 and storage has been allocated. BAD will contain the FWA/100_8 (relative to RA) of the assigned buffer. IODC resets BAD to zero when BAF=0 and storage has been released.

c) DCB = Director Call Byte. DCB is altered by EXEC and FNTS and is not used by IODC.



RTD = ReTurn Done flag. RTD is set when a call to EXEC has accomplished the desired task. RTD will also be set by FNTS if the processor has called IDLE and a) if this is the message processor and a message is available from the central site operator, or b) if this is the print processor, MAD=0 (see below), and a file is available for printing. RTD, being set on a call to IDLE, will cause an immediate return to the processor. RTD is returned to the processor.

RTN = ReTurn Not-ready flag. RTN is set when a call to EXEC for I/O via IODC has produced a not-ready status on the requested device. RTN is returned to the processor.

RTE = ReTurn Error flag. RTE is set when a call to EXEC to read cards has produced a bad-card-code error from IODC. RTE is returned to the processor.

ERQ = EXEC call ReQuest. ERQ is set by EXEC to indicate the type of call made. The values are:

1 = WAIT

2 = MESG

3 = STAL

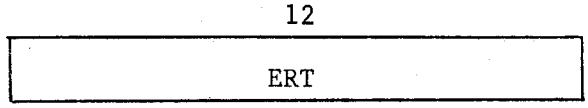
4 = PRNT

5 = READ

6 = IDLE

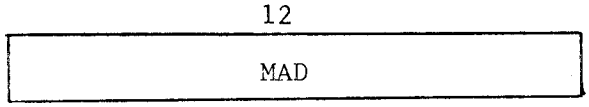
10_8 = EXEC (message processor only)

d) ERT = EXEC ReTurn address. ERT is set by EXEC and is not used by IODC.



ERT is the return address for the last call to EXEC made by the processor.

e) MAD = Message ADdress. MAD is set by EXEC and FNTS and is not used by IODC.



MAD is set by EXEC to the low order 12 bits of the A register when the processor calls IDLE, MESG, or STAL. MAD will be set by FNTS if this is the print processor director, and if the current call is to IDLE, and if MAD=RTD=0, and if a print file is available for printing. In this case, MAD will be set to the address of the FNT entry for the print file, and RTD will be set to 1. MAD is returned to any processor when a return is made from an IDLE call. For a MESG or STAL call, MAD is the PP address of the message to be displayed. If MAD=0, the message will be sent from the circular buffer whose address is in BAD. Thus, for STAL where BAD=0, only a +blank will be sent if MAD=0. MAD is not returned to the processor after a return from a MESG or STAL call.

2) CLPC = Current Line Processor storage for the Current processor. CLPC is used only by the processor and is neither used nor altered by IODC or EXEC. CLPC is the name of the second word of the processor storage area of the currently active processor. SAV1 through SAV5 are used by the processor to save internal flags and addresses during calls to EXEC subroutines.

2.3 Details of IODC and EXEC Routines (1EU)

2.3.1 Narrative of IODC and EXEC Routines

PROGRAM - 1EU
ROUTINE - INIT

INIT is the first routine to be executed when 1EU is loaded. INIT partially initializes the PP direct cells for IODC, and attempts to enter any new lines into the string of assigned lines. When new lines are ready to be entered, INIT scans the entire line status table and examines those lines which have been assigned to EXPORT/IMPORT 200 but not entered into the EXPORT/IMPORT 200 string. If the line is a CRT, then the line is initialized and entered into the string; otherwise, the line is re-assigned to subsystem zero. A dayfile message is issued to indicate the time at which each line is assigned.

PROGRAM - 1EU
ROUTINE - TERM

TERM is executed following INIT and performs the function of dropping the subsystem and/or individual lines. If the central site operator has requested a drop {subsystem drop} and there are no lines in the E/I 200 string, then TERM will drop the subsystem, delete its availability, and drop the PP.

If a line drop has been requested, TERM scans the EXPORT string. Each line in the string that is ready to be dropped is reassigned to subsystem zero, and the other lines in the string are relinked to bypass the dropped lines. A dayfile message is issued to indicate the time at which each line is dropped.

PROGRAM - 1EU
ROUTINE - FNTS

FNTS is executed following TERM. The primary tasks of FNTS are, 1) to process messages from the central site operator and, 2) to periodically scan the FNT. When a message has been entered by the central site operator, FNTS will scan the string of E/I 200 lines and either route the message to the appropriate terminal or issue a dayfile message indicating the specified terminal ID was invalid. In the case of a message to all terminals, FNTS will pre-scan all assigned lines. If all terminals have received the message, FNTS will clear the central site request. Otherwise FNTS will route the message to all terminals. Scanning of the FNT is performed no faster than once each second. When it is time to scan the FNT, FNTS first scans the string of E/I 200 lines and produces a table that contains an entry for each terminal that is ready to receive a new print file. The FNT is then pre-scanned to determine if any E/I 200 output files exist. If there are files, FNTS assigns the FNT/FST channels and does a complete FNT scan. All E/I 200 output files which are not print files are diverted to the central site, while print files are routed, if possible, to the terminal that submitted the job. The routing information is saved in the internal table of ready terminals and is overwritten with the routing information of higher priority print files as they are encountered.

PROGRAM - 1EU
ROUTINE - FNTS {Continued}

When the FNT scan is completed, the table of ready terminals is scanned and any routing information that was stored is used to connect the print file to its respective terminal. The FNT/FST channels are then released and IODC is entered.

PROGRAM - 1EU
ROUTINE - I0DC

I0DC is entered following FNTS and serves as the main link between I0D and E/I 200. To the higher level routines in E/I 200 (namely, EXEC and the processors), I0DC is a subroutine which is called to perform I/O on a single line. The primary duties of I0DC are to, 1) cycle through all the lines in the string, 2) set up the I/O calls to I0D, 3) request retries on error conditions, and 4) suballocate certain items (such as storage and the central display) when they are requested by the lines. When I0DC is entered, it performs one pass through all the lines in the string and then a second pass if certain conditions were detected during the first pass. For each line in the string, I0DC performs as follows.

If the line is actively performing I/O, no return will be made to the calling routine but instead I0DC will check to see if the line is only waiting on a poll. If this is true and there is some reason to stop the polling (generally the end of a two-second polling period), I0DC will request I0D to stop polling. In either case, I0DC will then bypass the return section and proceed to the sub-allocation section. If the line is not actively

PROGRAM - 1EU
ROUTINE - I0DC {Continued}

performing I/O, then I0DC enters the return section. In the return section, I0DC examines the control information returned by I0D, and the request made by the calling routine. From this information, a return code is constructed and I0DC returns control to the calling routine {EXEC}. Exceptions to the above are:

If a line drop is in progress for this line but has not been completed by TERM, I0DC will bypass the return and go to the sub-allocation section.

If the line is in the process of logging out but EXEC has not yet requested a drop, I0DC interprets the call and returns a LOGOUT command for each call that requested a poll.

If a hardware error occurred, I0DC will request I0D to retry the transmission. The remainder of the return section will be bypassed and I0DC will go to the sub-allocation section.

PROGRAM - 1EU
ROUTINE - I0DC {Continued}

When I0DC is called again, the I/0 request specified by the call is translated into the information necessary for I0D, and I/0 for this line is activated. The only exceptions are if a call to drop the line was made, or if the line is in the process of logging out. In these cases, no I/0 is activated. Following the return section, I0DC enters the sub-allocation section. If the line desires to write to the central display, I0DC will first allocate the display to E/I 200 and then to the line. The allocation is on a first come basis and is generally processed in two or more passes through the line string. For central memory allocation, I0DC searches through the line's three processors to look for requests, and then assigns or releases storage in a manner similar to that for the central display. A somewhat more complex checking scheme is included, however, to take care of the possible assign-release conflicts that can arise. After the sub-allocation section is completed, I0DC cycles on to the next line in the string. When the string is exhausted, I0DC will make a second pass if storage was assigned during the first pass so as to reduce waiting-for-storage time at the terminal. Finally, after all

PROGRAM - 1EU
ROUTINE - I0DC {Continued}

passes are complete, I0DC updates the subsystem status information and drops the PP.

PROGRAM - 1EU
ROUTINE - EXEC

EXEC is a set of re-entrant subroutines that act as an interface between the processors and IODC. With the exception of three of the subroutines, EXEC processes the calls in a cyclic fashion so that each processor is serviced once in each cycle. At the end of each cycle, EXEC examines tha status of all three processors and issues the idle message {time, date, and terminal ID} if all the processors are idle. While in this idle state, EXEC will request the line to be dropped if a logout was in progress, or will periodically check to see if any of the processors should be reactivated. As soon as a processor is reactivated {and while one or more processors remain active} EXEC commences normal cycling of the processor calls. Should a processor receive a command from the remote operator, EXEC stops cycling, and "locks onto" the specified processor until the processor requests the command be released. No I/O can be performed during the period, and any calls to EXEC subroutines are treated as nops. EXEC automatically takes care of those cases where a call to IODC for one processor produces a command for another, by reissuing the IODC call until the I/O is complete, or a command is received for the requesting processor. The subroutines are described below {see

PROGRAM - 1EU
ROUTINE - EXEC {Continued}

section 2.1 for details of the calling sequences and return codes}.

EXEC - EXEC may be called only by the message processor and is used to perform an indirect call to I0DC. EXEC bypasses normal cycling for this subroutine.

CMRL - CMRL releases a processor from being locked onto a command. CMRL is performed within EXEC and no call is made to I0DC. EXEC bypasses normal cycling for this subroutine.

WAIT - WAIT provides a processor with a time delay of about 130ms to allow for completion of calls to other PP programs {such as CI0}. EXEC calls I0DC to perform a nop which provides no I/O but allows I0DC to continue through the E/I 200 string. EXEC bypasses normal cycling for this subroutine.

IDLE - IDLE is called when a processor has completed its tasks and has nothing else to do. Storage allocated to the processor will be released and the processor will be returned to when there is something for it to do.

PROGRAM - 1EU
ROUTINE - EXEC (Continued)

READ - READ will transfer a block of card data into a CM buffer from the remote card reader and provide the processor with the status of the reader. Control is returned to the calling processor when the I/O is completed, or when a command is entered for the processor.

PRNT - PRNT will transfer a block of print data from a CM buffer to the remote line printer and provide the processor with the status of the printer. Control is returned to the processor when the I/O is completed or when a command is entered for the calling processor.

MESG - MESG will display a message on the remote display. The first processor to call MESG (or STAL below) is "locked" onto the display until a command is entered for that processor. Other processors calling MESG (or STAL) will be held up until the command is received. If other processors are calling READ or PRNT, the message will be interrupted after two seconds to perform I/O for the other processors (as part of normal cycling) and then will be redisplayed for another two seconds. The message "lock", however, will not be altered by the interleaving. Control is returned to the calling processor when a

PROGRAM - 1EU
ROUTINE - EXEC {Continued}

command is entered for the processor.

STAL - STAL allocates storage for the processor and displays a message on the remote display during this period. The displayed message is treated exactly as for a call to MESG. Control is returned to the calling processor when storage is allocated, or when a command is entered for the processor.

Before returning to any of the processors, EXEC loads the upper-half of PP memory {initially containing INIT, TERM and FNTS} with the correct processor overlay. The processor overlays are also loaded when a message is to be re-displayed during a call to MESG or STAL. In this case, the overlay is required so that EXEC may obtain a new copy of the message to be displayed.

PROGRAM - 1EU
ROUTINE - CTFL

CTFL is a subroutine that calls R.TFL to check CM addresses, relative to RA, and then add RA to the given address. If the address is outside the field length, the system failure procedure is followed (Section 11.1) and the PP hangs until an operator drop is issued. An entry point to CTFL is also provided for for the processor overlays that are loaded into the PP.

PROGRAM - 1EU
ROUTINE - LTIM

LTIM loads the PP memory with the current value of the IOD system clock in 1/4's of seconds.

PROGRAM - 1EU
ROUTINE - OVLY

OVLY is used to load the processor overlays when 1) they are returned to and 2) when messages contained within them are to be re-displayed. OVLY calls R.OVL to load the overlays.

PROGRAM - 1EU
ROUTINES - LSSR, LSTS, LCLS, LCLR, LLDP, LCLD, LCLP, LCIE, LCLM

These routines are used to read words from the subsystem table entry or the current line table entry into the PP.

PROGRAM - 1EU
ROUTINES - SSSR, SSTS, SCLS, SCLR, SLCP, SCLD, SCLP, SCIE, SCLM

These routines are used to write words to the subsystem table entry or the current line table entry from the PP.

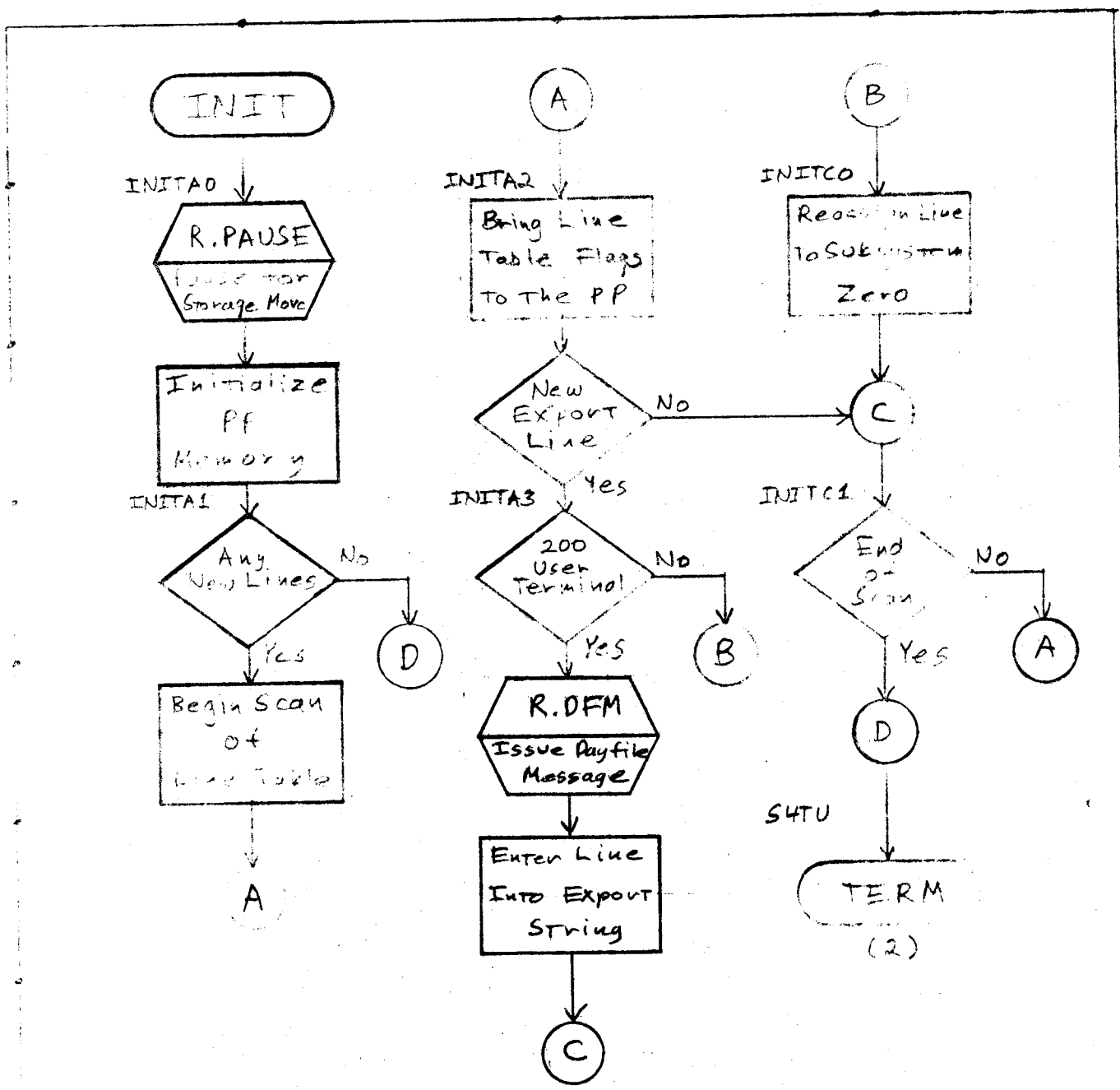
PROGRAM - 1EU
ROUTINE - CJCB

CJCB generates the terminal ID (job characters 4 and 5) for INIT, TERM, FNTS, and IODC. The first character of the ID (job character 4) is computed directly from the site address (SA). The second ID character (job character 5) is found from a table that is created at assembly time by comparing entries with the port and multiplexor numbers (MX and PN).

2.3.2 Flow Chart of IODC and EXEC Routines

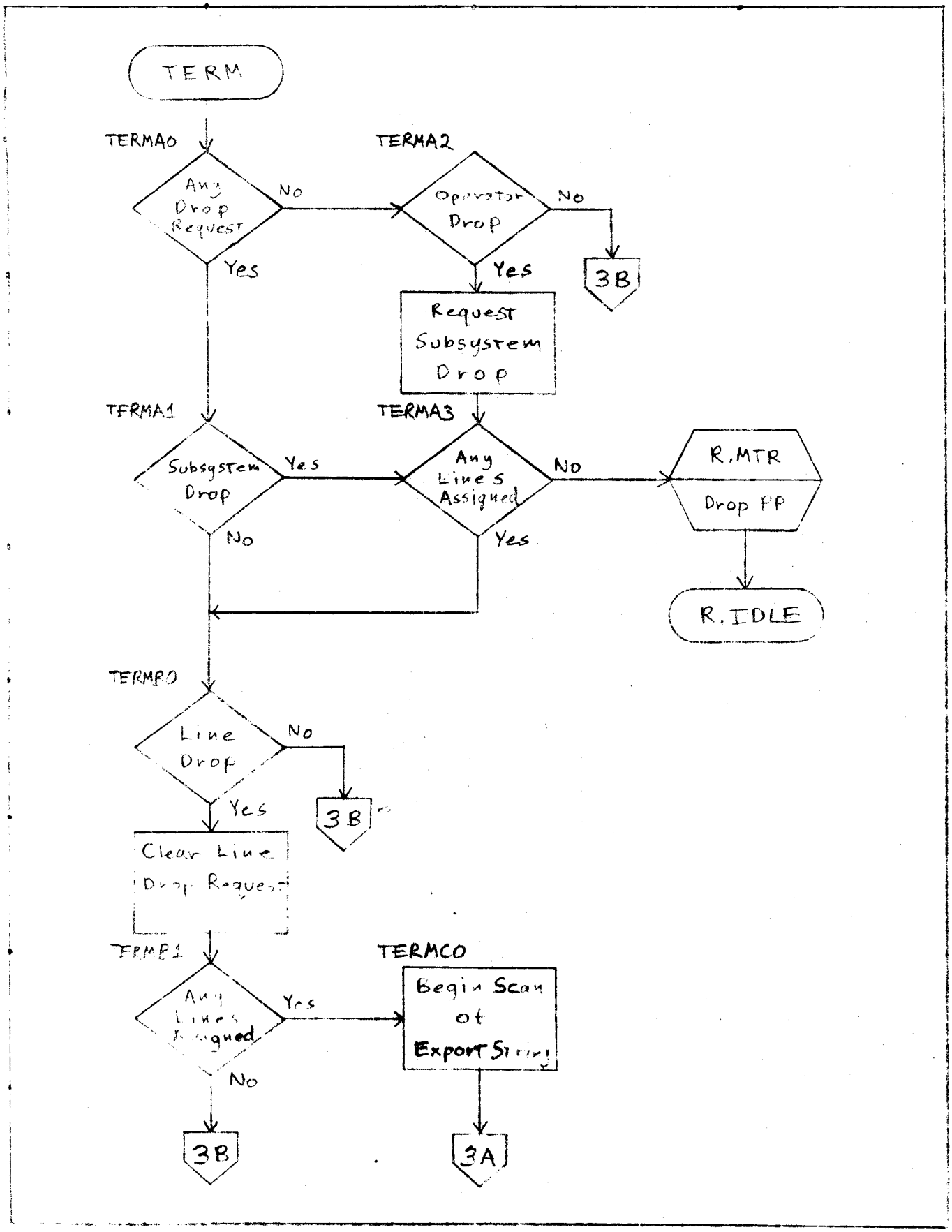
1EU

Page 1


INIT
INITA0
R.PAUSE
Pause for Storage Move
Initialize PP Memory
INITA1
Any New Lines
No
Yes
D
Begin Scan of Line Table
A
A
INITA2
Bring Line Table Flags To The PP
New Export Line
No
Yes
INITA3
200 User Terminal
No
Yes
B
R.DFM
Issue Dayfile Message
Enter Line Into Export String
C
B
INITC0
Reassign Line To Subsystem Zero
C
INITC1
End of Scan
No
Yes
A
D
S4TU
TERM
(2)

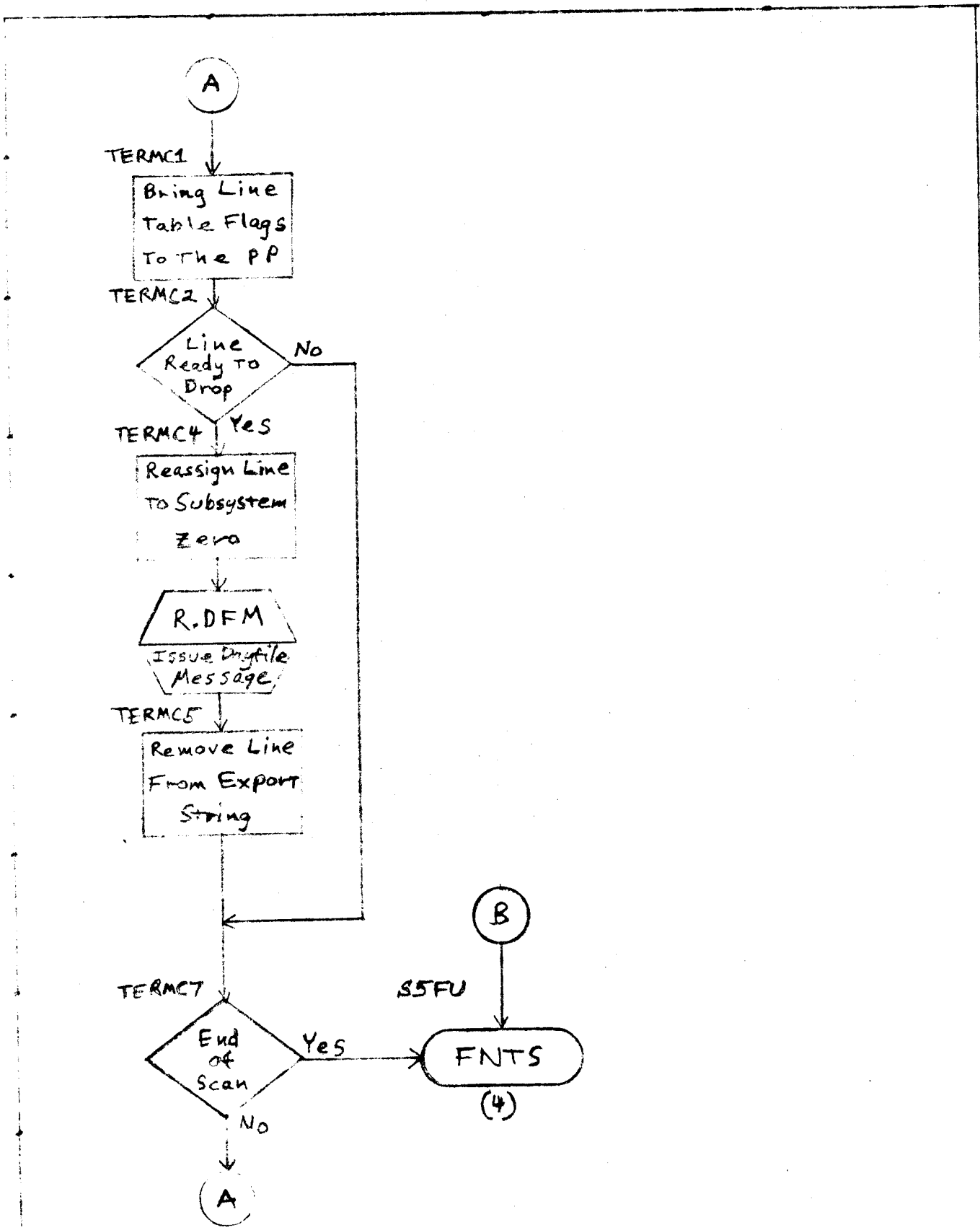
1F1)

Page 2

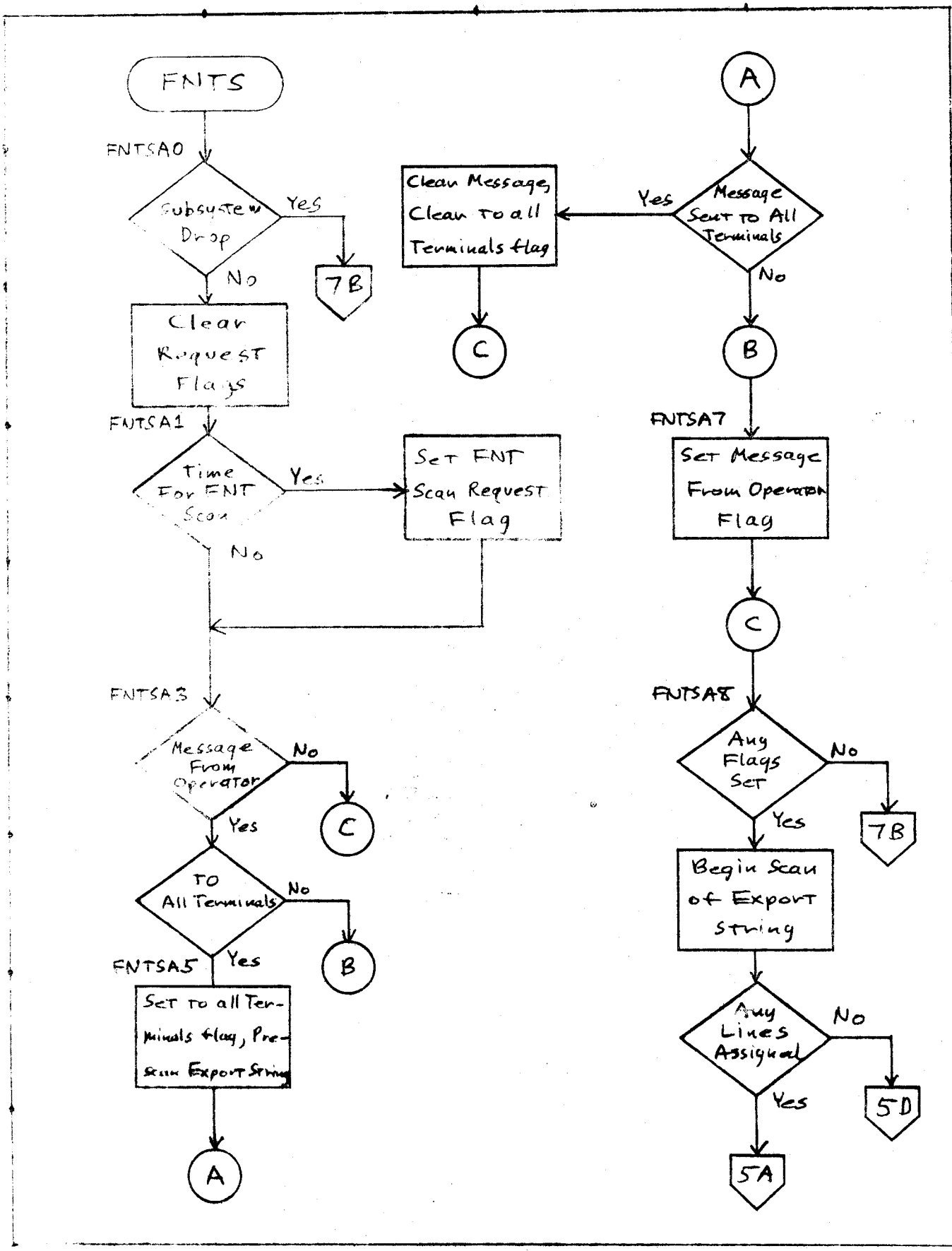

TERM
TERMA0
Any Drop Request
No
Yes
TERMA2
Operator Drop
No
Yes
3B
Request Subsystem Drop
TERMA1
Subsystem Drop
Yes
No
TERMA3
Any Lines Assigned
No
Yes
R.MTR
Drop PP
R.IDLE
TERMB0
Line Drop
No
Yes
3B
Clear Line Drop Request
TERMB1
Any Lines Assigned
Yes
No
3B
TERMC0
Begin Scan of Export String
3A

1EU

Page 3

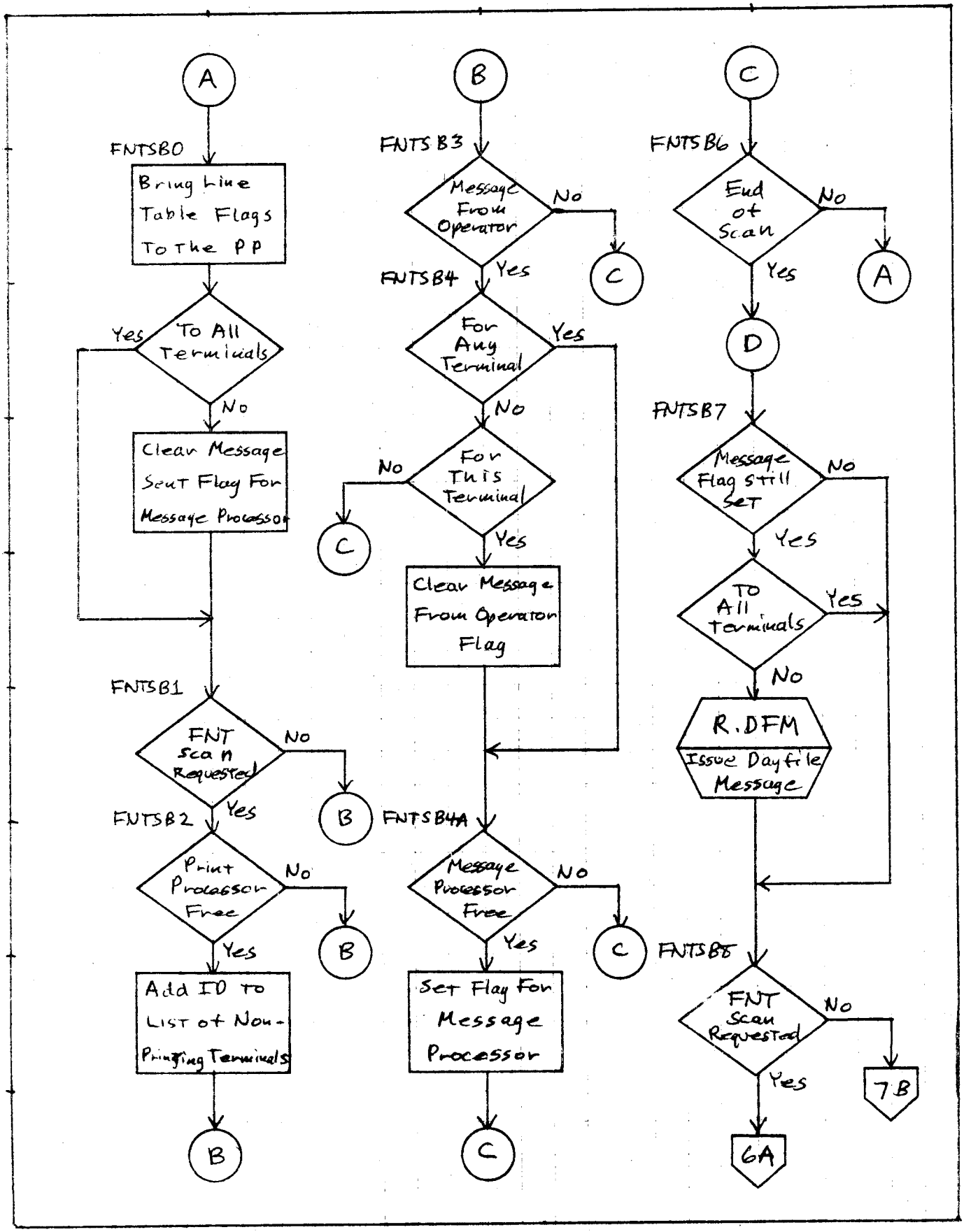

A
TERMC1
Bring Line Table Flags To The PP
TERMC2
Line Ready To Drop
No
Yes
TERMC4
Reassign Line To Subsystem Zero
R.DFM
Issue Dayfile Message
TERMC5
Remove Line From Export String
B
S5FU
TERMC7
End of Scan
Yes
FNTS
(4)
No
A

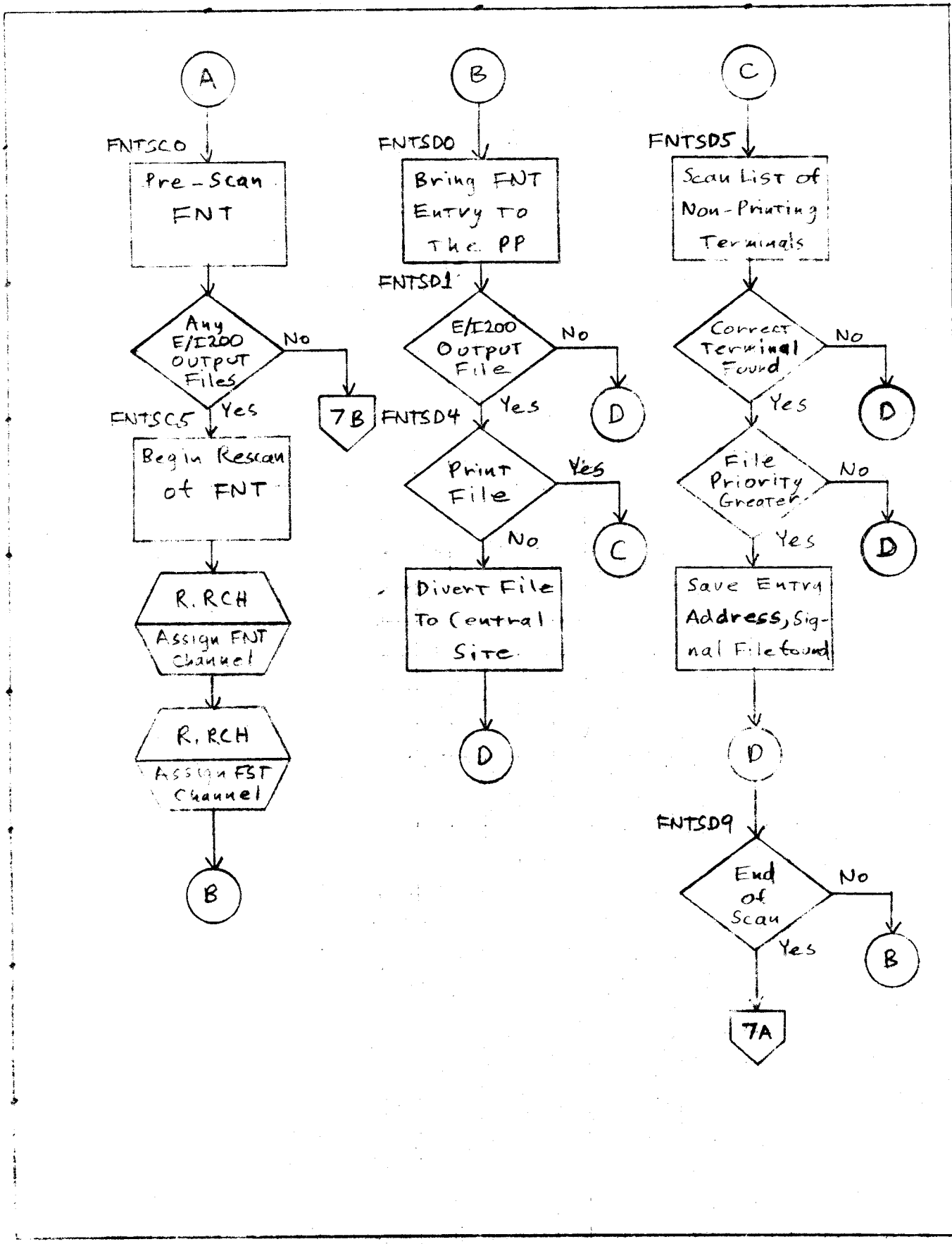
1 EU


FNTS
FNTSA0
Subsystem Drop
Yes
No
7B
Clear Request Flags
FNTSA1
Time For ENT Scan
Yes
No
Set ENT Scan Request Flag
FNTSA3
Message From Operator
No
Yes
C
TO All Terminals
No
Yes
B
FNTSA5
Set to all Terminals flag, Pre-scan Export String
A
A
Message Sent to All Terminals
Yes
No
Clear Message, Clear to all Terminals flag
C
B
FNTSA7
Set Message From Operator Flag
C
FNTSA8
Any Flags Set
No
Yes
7B
Begin Scan of Export String
Any Lines Assigned
No
Yes
5D
5A

1EU

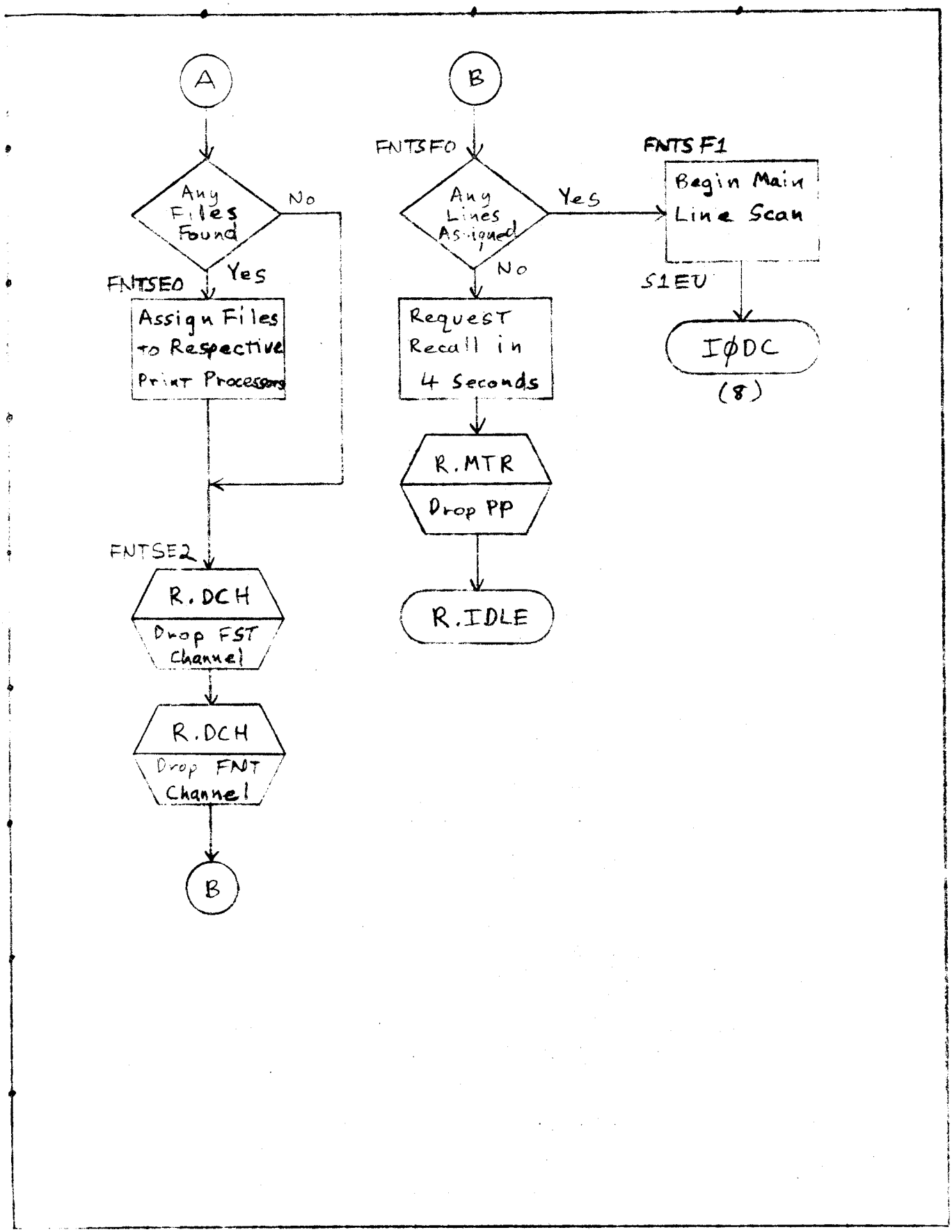
Page 5


A
FNTSB0
Bring Line Table Flags To The PP
To All Terminals
Yes
No
Clear Message Sent Flag For Message Processor
FNTSB1
FNT scan Requested
No
Yes
B
FNTSB2
Print Processor Free
No
Yes
B
Add ID to List of Non-Printing Terminals
B
B
FNTSB3
Message From Operator
No
Yes
C
FNTSB4
For Any Terminal
Yes
No
For This Terminal
No
Yes
C
Clear Message From Operator Flag
FNTSB4A
Message Processor Free
No
Yes
C
Set Flag For Message Processor
C
C
FNTSB6
End of Scan
No
Yes
A
D
FNTSB7
Message Flag Still Set
No
Yes
TO All Terminals
Yes
No
R.DFM
Issue Dayfile Message
FNTSB8
FNT Scan Requested
No
Yes
7B
6A


A
FNTSC0
Pre-Scan FNT
Any E/I200 OUTPUT Files
No
Yes
FNTSC5
7B
Begin Rescan of FNT
R.RCH
Assign FNT Channel
R.RCH
Assign FST Channel
B
B
FNTSD0
Bring FNT Entry To The PP
FNTSD1
E/I200 OUTPUT File
No
Yes
D
FNTSD4
Print File
Yes
No
C
Divert File To Central Site
D
C
FNTSD5
Scan List of Non-Printing Terminals
Correct Terminal Found
No
Yes
D
File Priority Greater
No
Yes
D
Save Entry Address, Signal File found
D
FNTSD9
End of Scan
No
Yes
B
7A

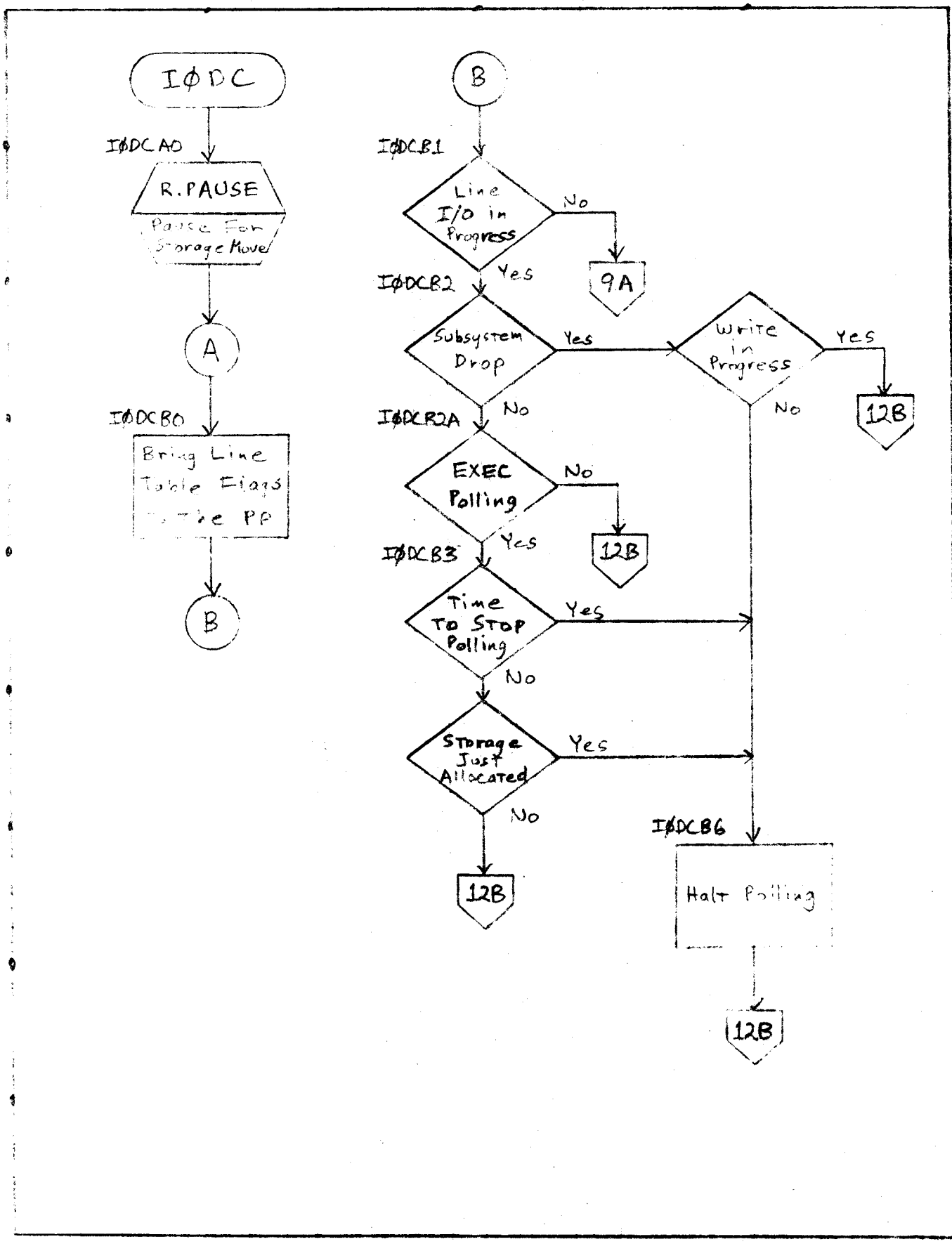
1EU

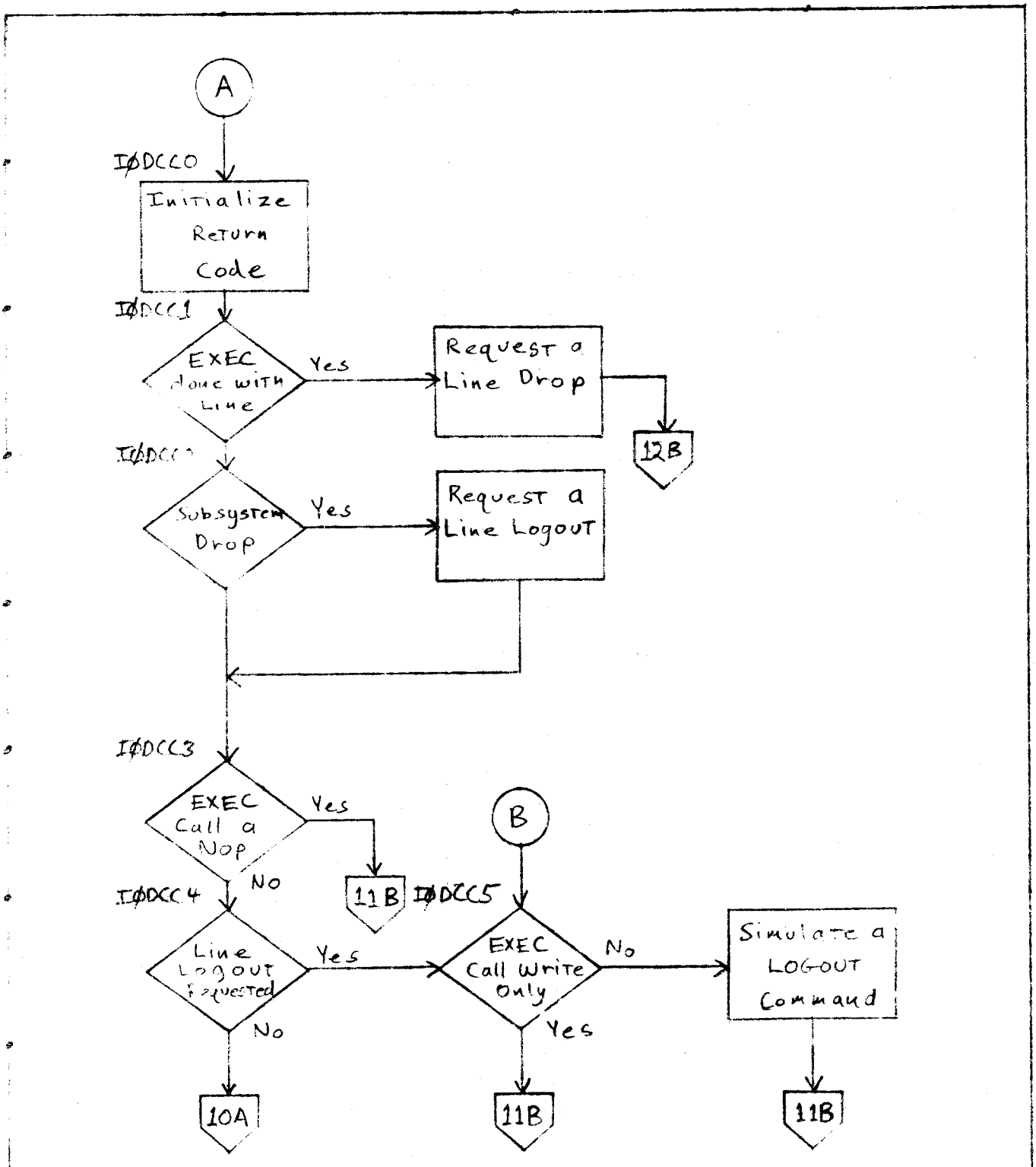
Page 7


A
Any Files Found
No
Yes
FNTSE0
Assign Files to Respective Print Processors
FNTSE2
R.DCH
Drop FST Channel
R.DCH
Drop FNT Channel
B
B
FNTSF0
Any Lines Assigned
Yes
No
FNTSF1
Begin Main Line Scan
S1EU
IØDC
(8)
Request Recall in 4 Seconds
R.MTR
Drop PP
R.IDLE

1FU

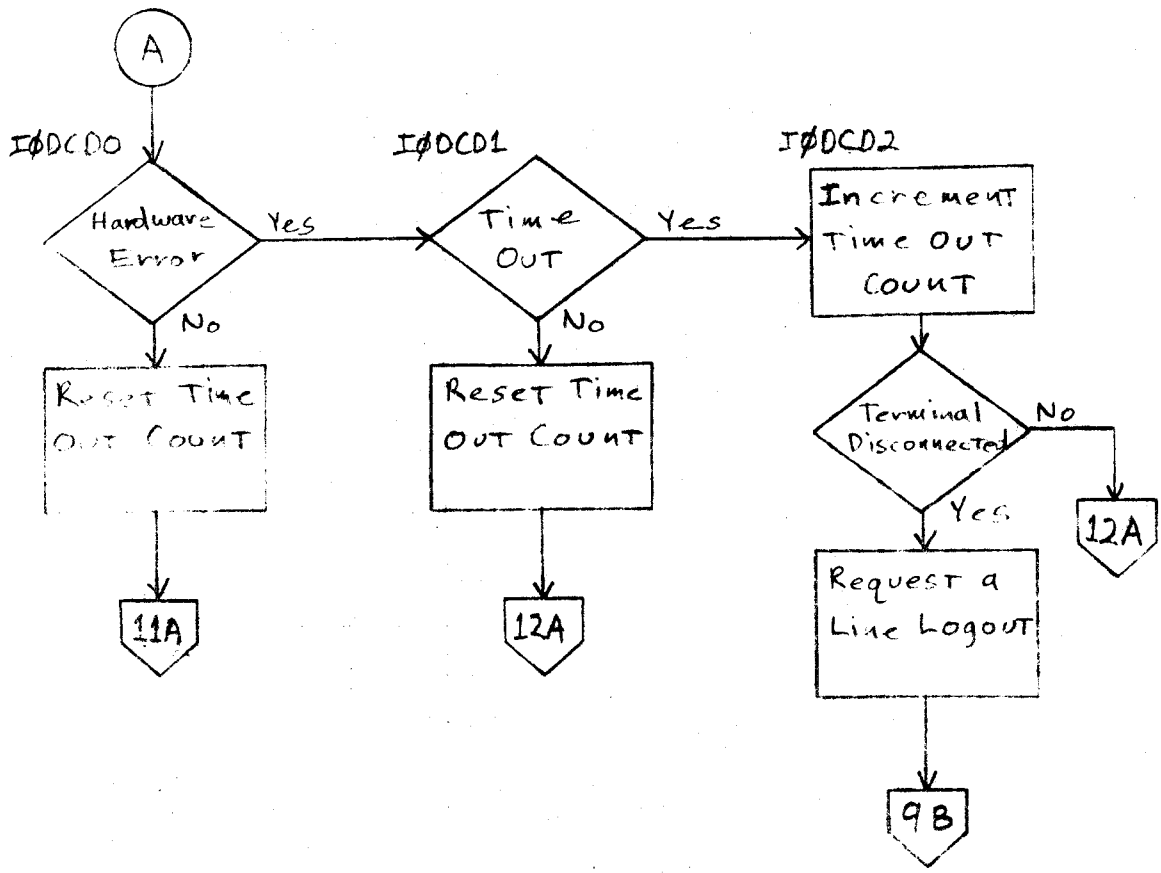
Page 8

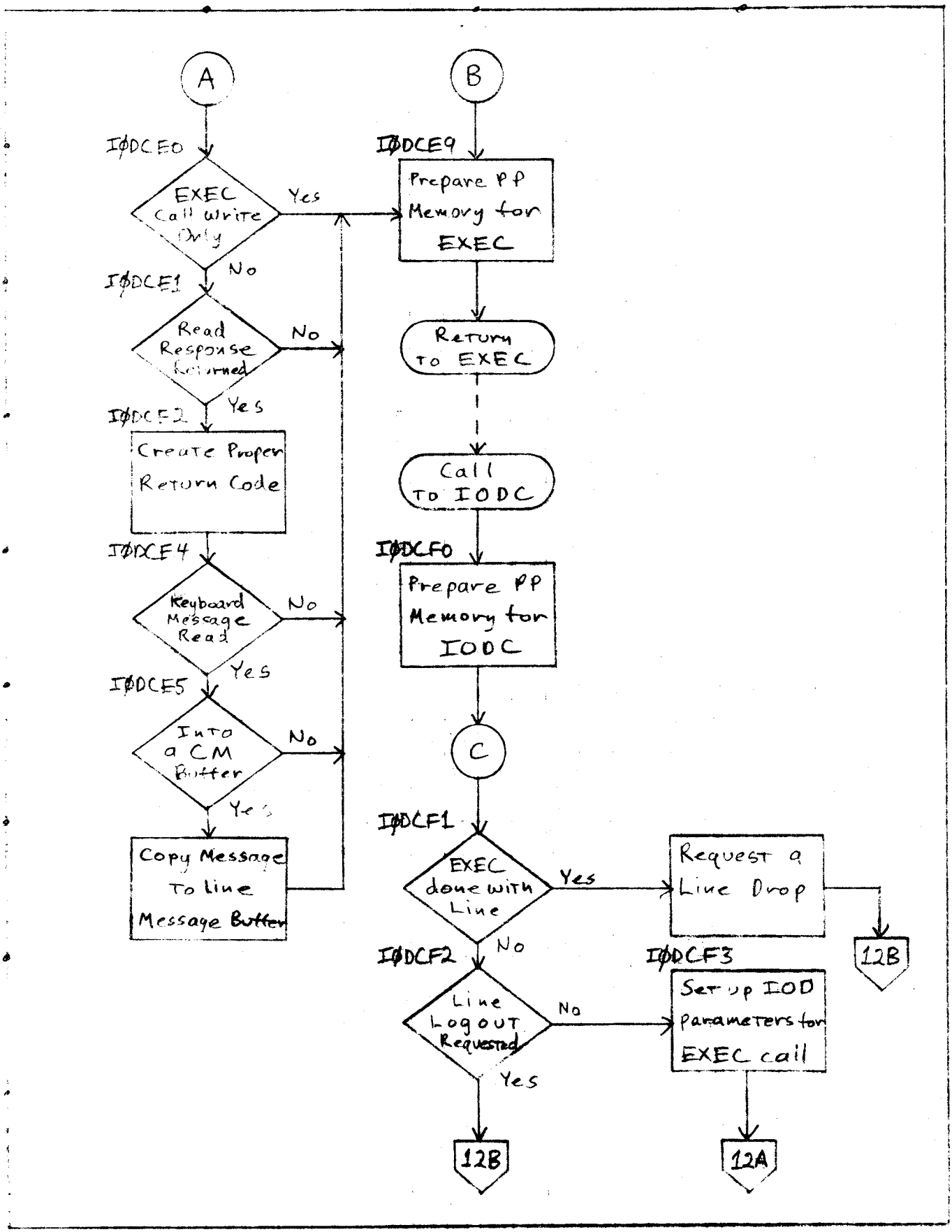

IØDC
IØDCA0
R.PAUSE
Pause For Storage Move
A
IØDCB0
Bring Line Table Flags to the PP
B
B
IØDCB1
Line I/O in Progress
No
9A
Yes
IØDCB2
Subsystem Drop
Yes
Write in Progress
Yes
12B
No
No
IØDCB2A
EXEC Polling
No
12B
Yes
IØDCB3
Time To Stop Polling
Yes
No
Storage Just Allocated
Yes
No
12B
IØDCB6
Halt Polling
12B

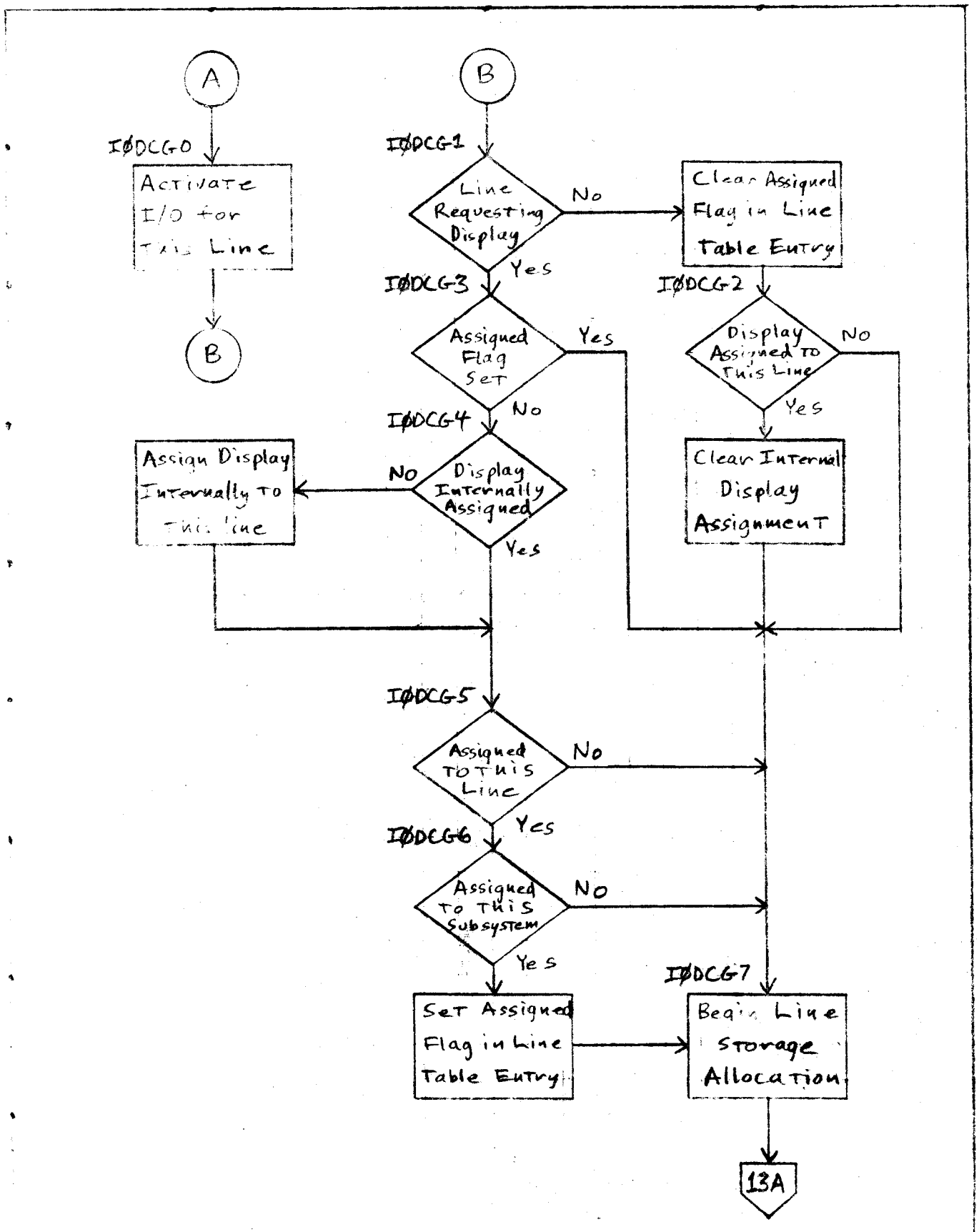

A
IØDCC0
Initialize Return Code
IØDCC1
EXEC done with Line
Yes
Request a Line Drop
12B
IØDCC2
Subsystem Drop
Yes
Request a Line Logout
IØDCC3
EXEC Call a Nop
Yes
No
11B
B
IØDCC4
Line Logout Requested
Yes
No
IØDCC5
EXEC Call Write Only
No
Yes
Simulate a LOGOUT Command
10A
11B
11B

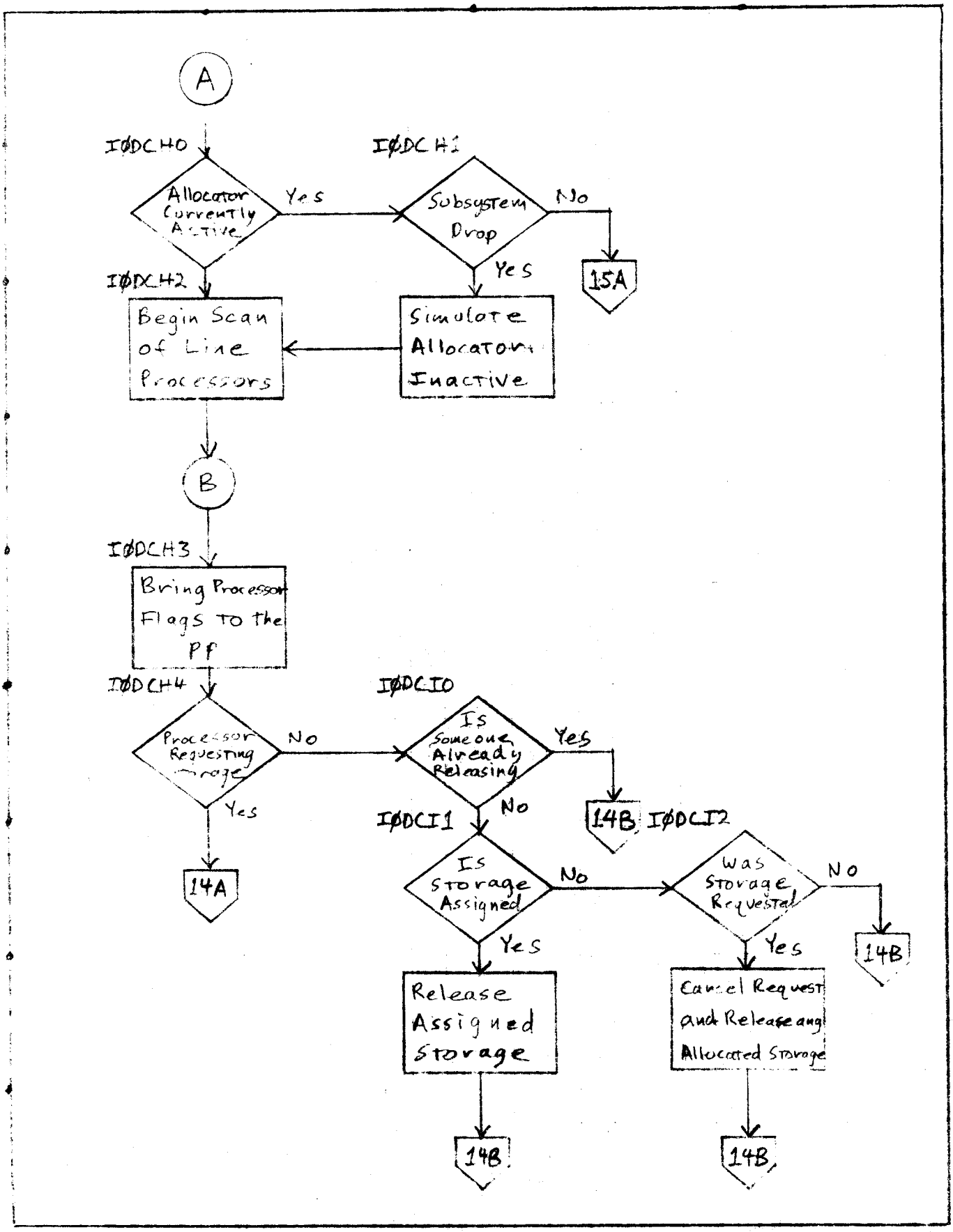
IEU

Page 10


A
IØDCD0
Hardware Error
Yes
No
Reset Time Out Count
11A
IØDCD1
Time Out
Yes
No
Reset Time Out Count
12A
IØDCD2
Increment Time Out Count
Terminal Disconnected
No
12A
Yes
Request a Line Logout
9B

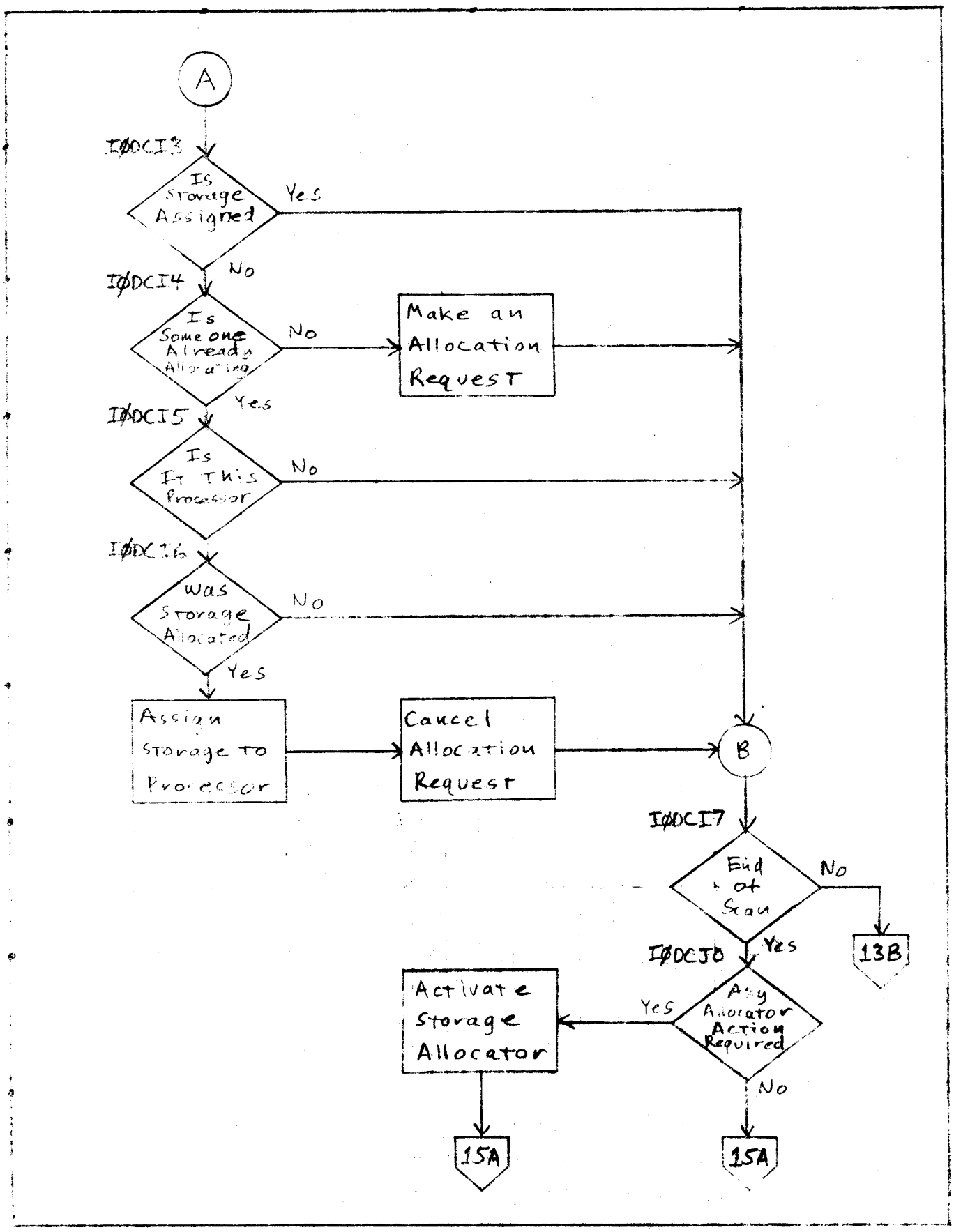

A
B
IØDCE0
EXEC Call Write Only
Yes
No
IØDCE1
Read Response Returned
No
Yes
IØDCE2
Create Proper Return Code
IØDCE4
Keyboard Message Read
No
Yes
IØDCE5
Into a CM Buffer
No
Yes
Copy Message To line Message Buffer
IØDCE9
Prepare PP Memory for EXEC
Return to EXEC
Call to IODC
IØDCF0
Prepare PP Memory for IODC
C
IØDCF1
EXEC done with Line
Yes
Request a Line Drop
12B
No
IØDCF2
Line Logout Requested
No
Yes
IØDCF3
Set up IOD parameters for EXEC call
12B
12A

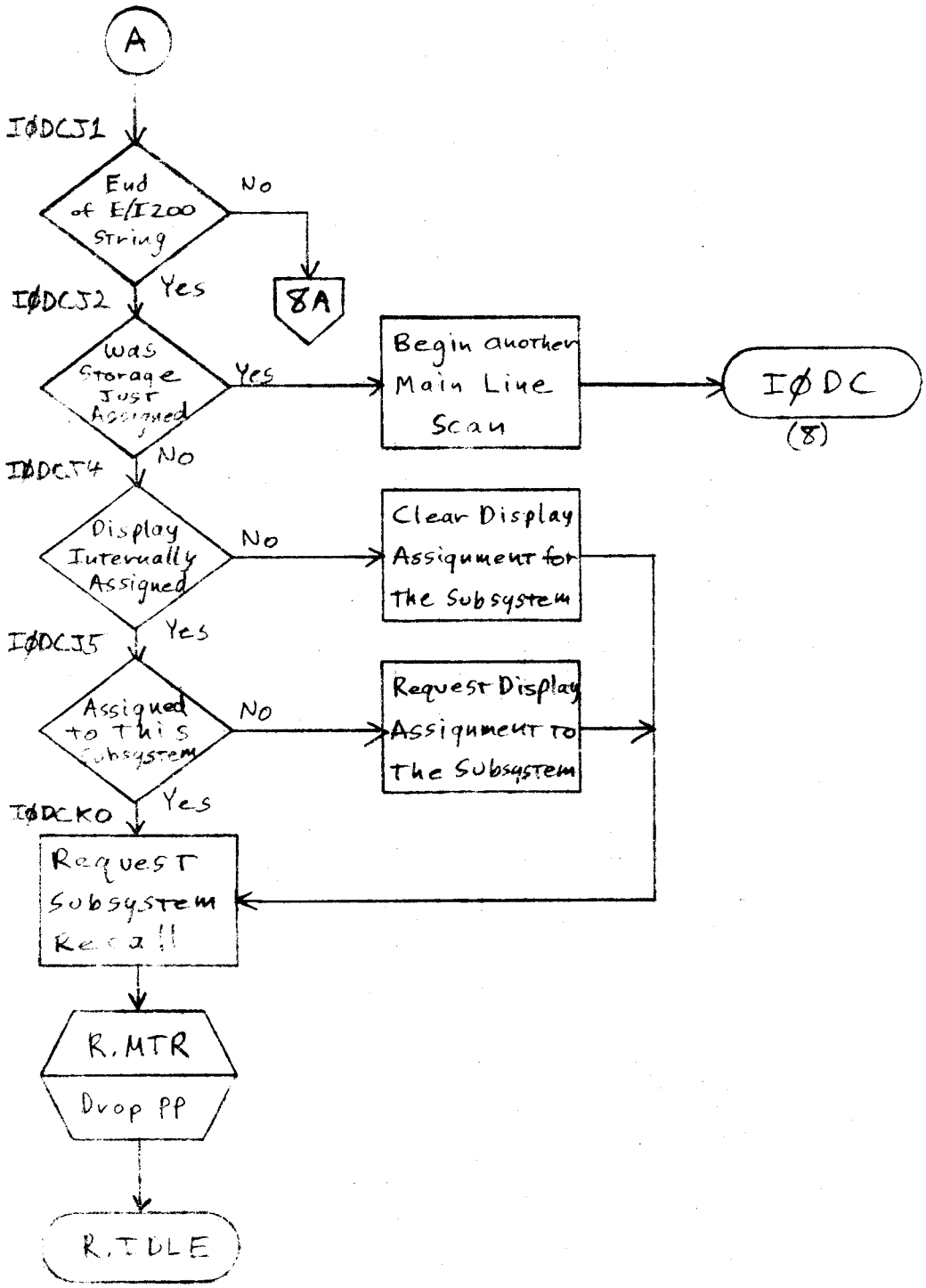

A
B
IØDCG0
Activate I/O for This Line
B
IØDCG1
Line Requesting Display
No
Clear Assigned Flag in Line Table Entry
Yes
IØDCG3
Assigned Flag Set
Yes
IØDCG2
Display Assigned to This Line
No
Yes
No
IØDCG4
Assign Display Internally to This Line
No
Display Internally Assigned
Yes
Clear Internal Display Assignment
IØDCG5
Assigned to This Line
No
Yes
IØDCG6
Assigned to This Subsystem
No
Yes
IØDCG7
Set Assigned Flag in Line Table Entry
Begin Line Storage Allocation
13A

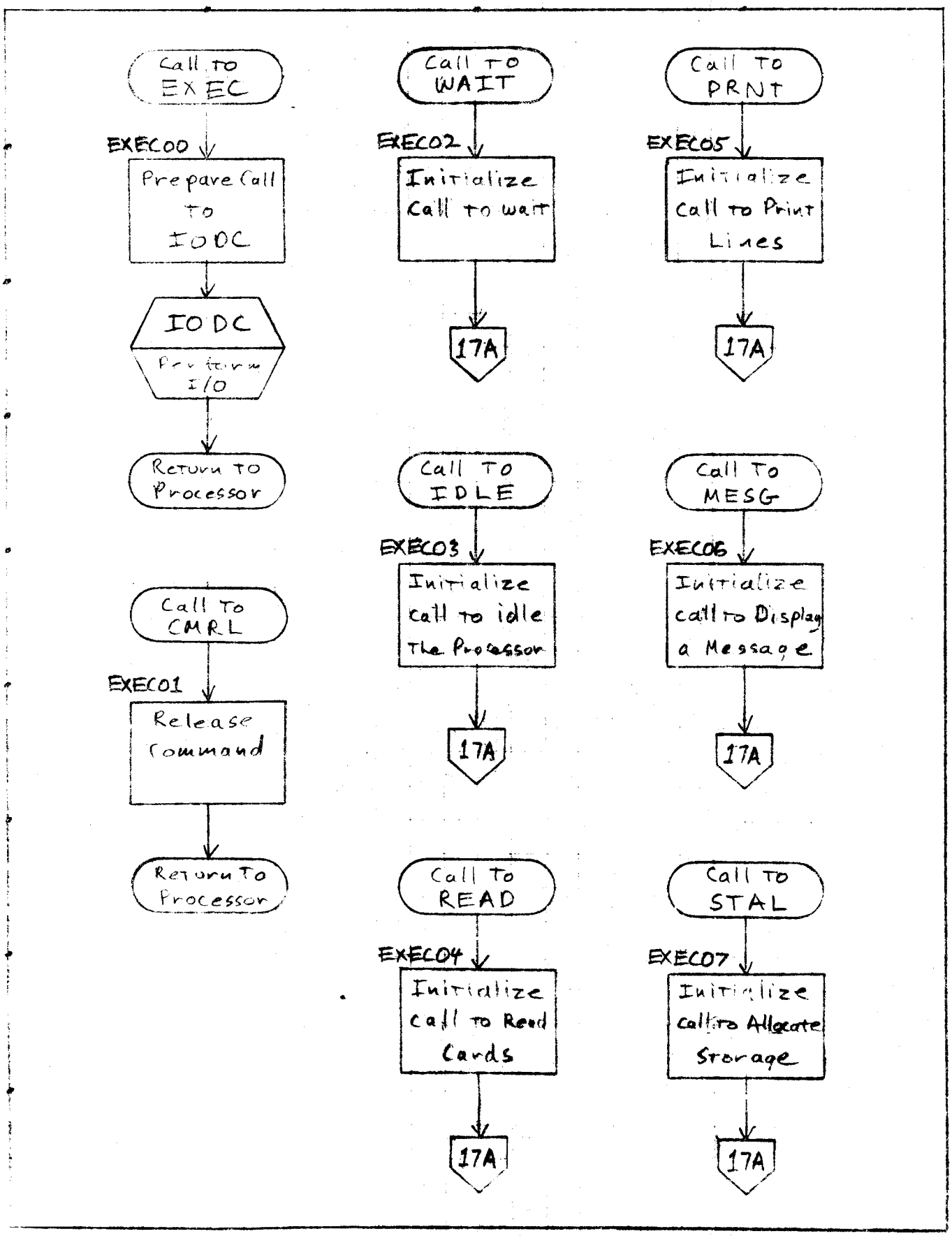

A
IØDCH0
Allocator Currently Active
Yes
IØDCH1
Subsystem Drop
No
15A
Yes
IØDCH2
Begin Scan of Line Processors
Simulate Allocator Inactive
B
IØDCH3
Bring Processor Flags to the PF
IØDCH4
Processor Requesting Storage
No
IØDCIO
Is Someone Already Releasing
Yes
14B
IØDCI2
Yes
14A
No
IØDCI1
Is Storage Assigned
No
Was Storage Requested
No
14B
Yes
Release Assigned Storage
Yes
Cancel Request and Release any Allocated Storage
14B
14B

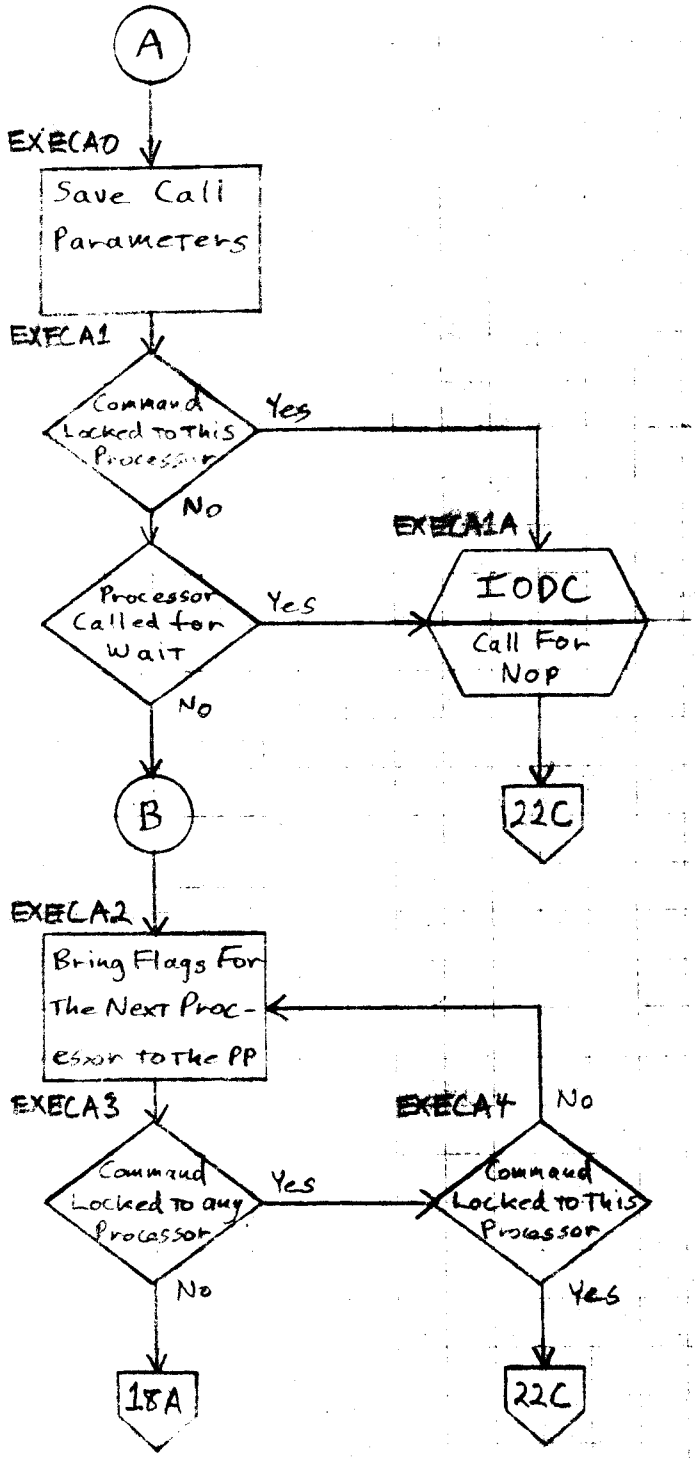
IFU

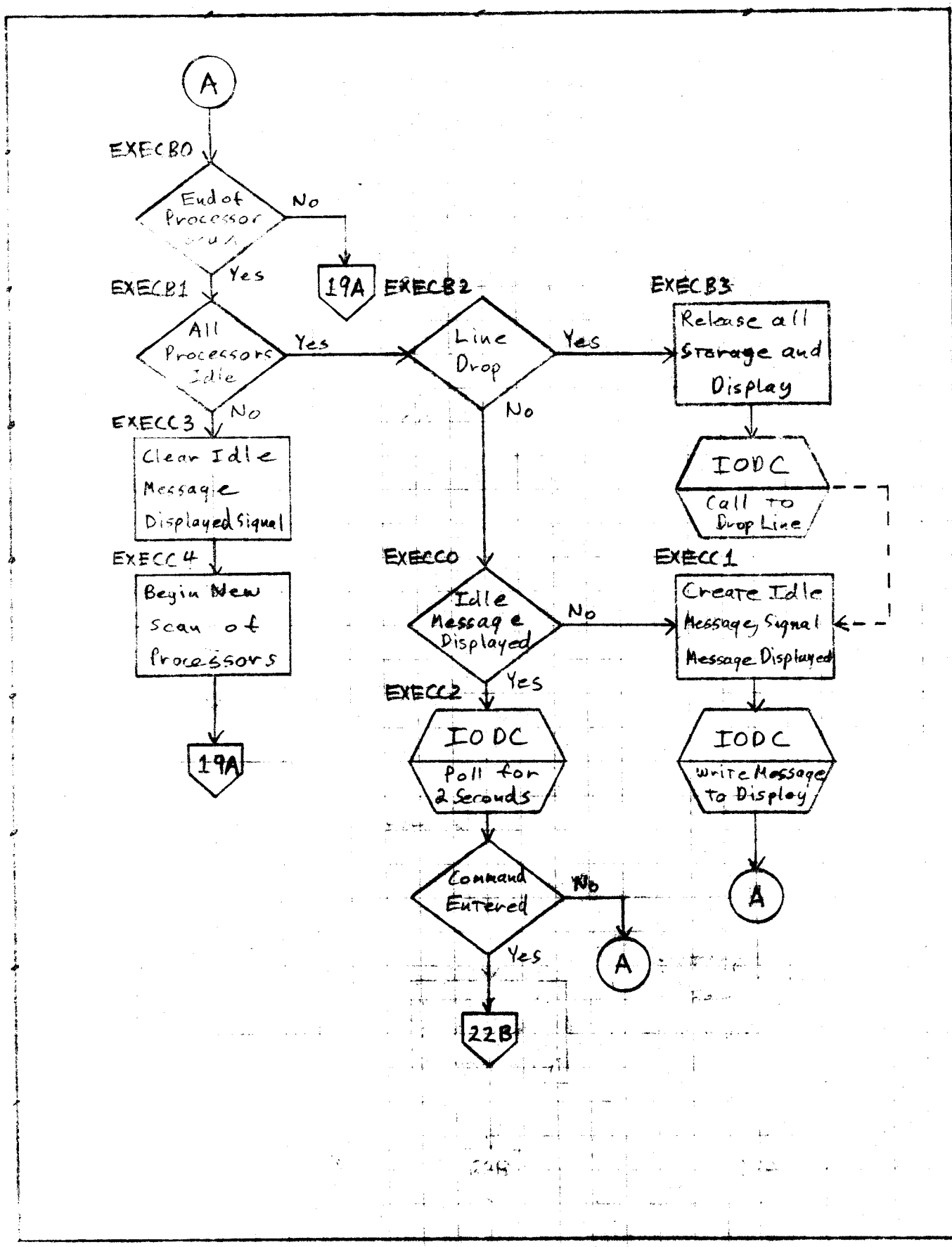
Page 14

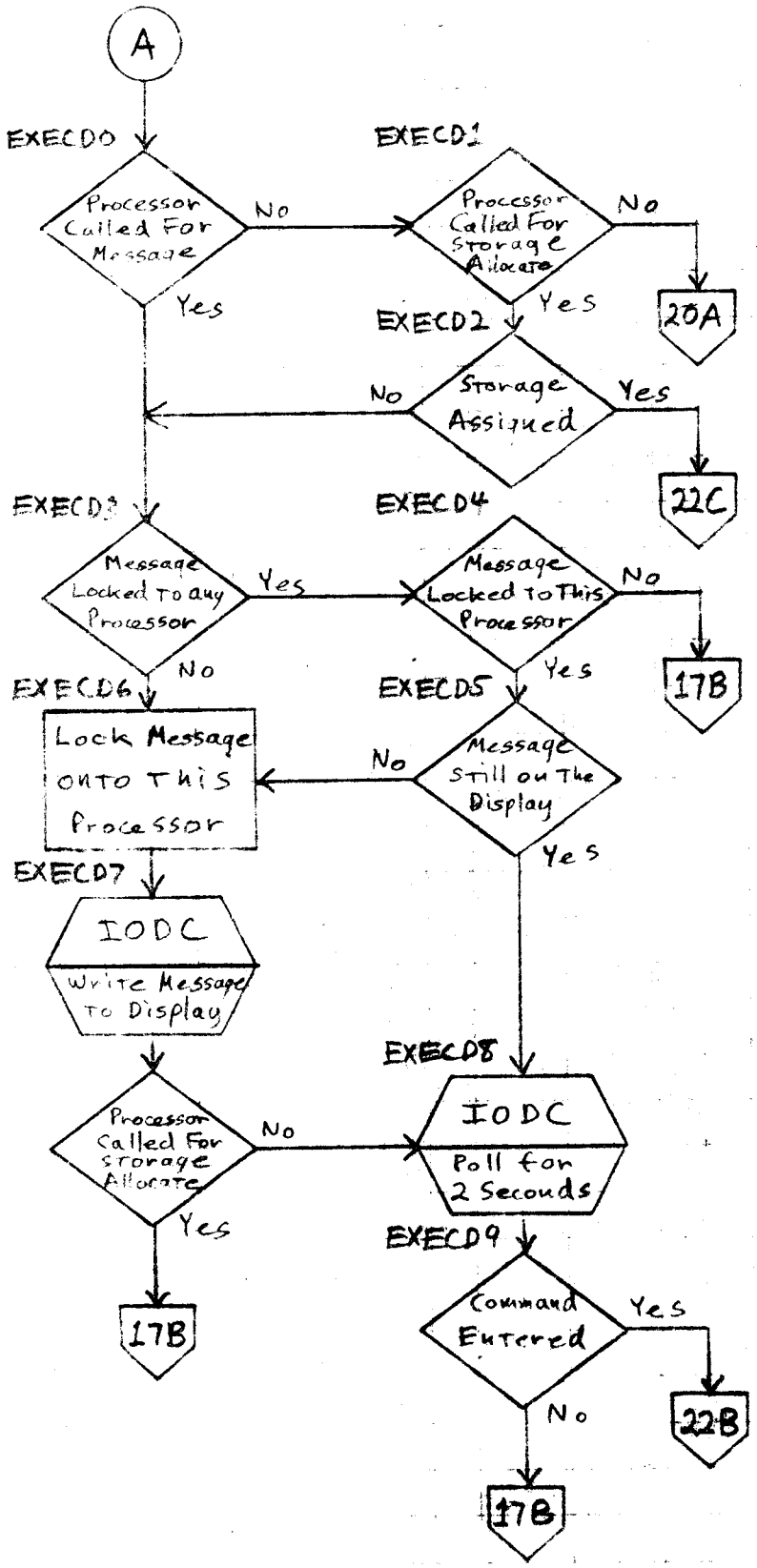

A
IØDCI3
Is Storage Assigned
Yes
No
IØDCI4
Is Someone Already Allocating
No
Make an Allocation Request
Yes
IØDCI5
Is It This Processor
No
IØDCI6
Was Storage Allocated
No
Yes
Assign Storage to Processor
Cancel Allocation Request
B
IØDCI7
End of Scan
No
13B
Yes
IØDCJ0
Any Allocator Action Required
Yes
Activate Storage Allocator
No
15A
15A

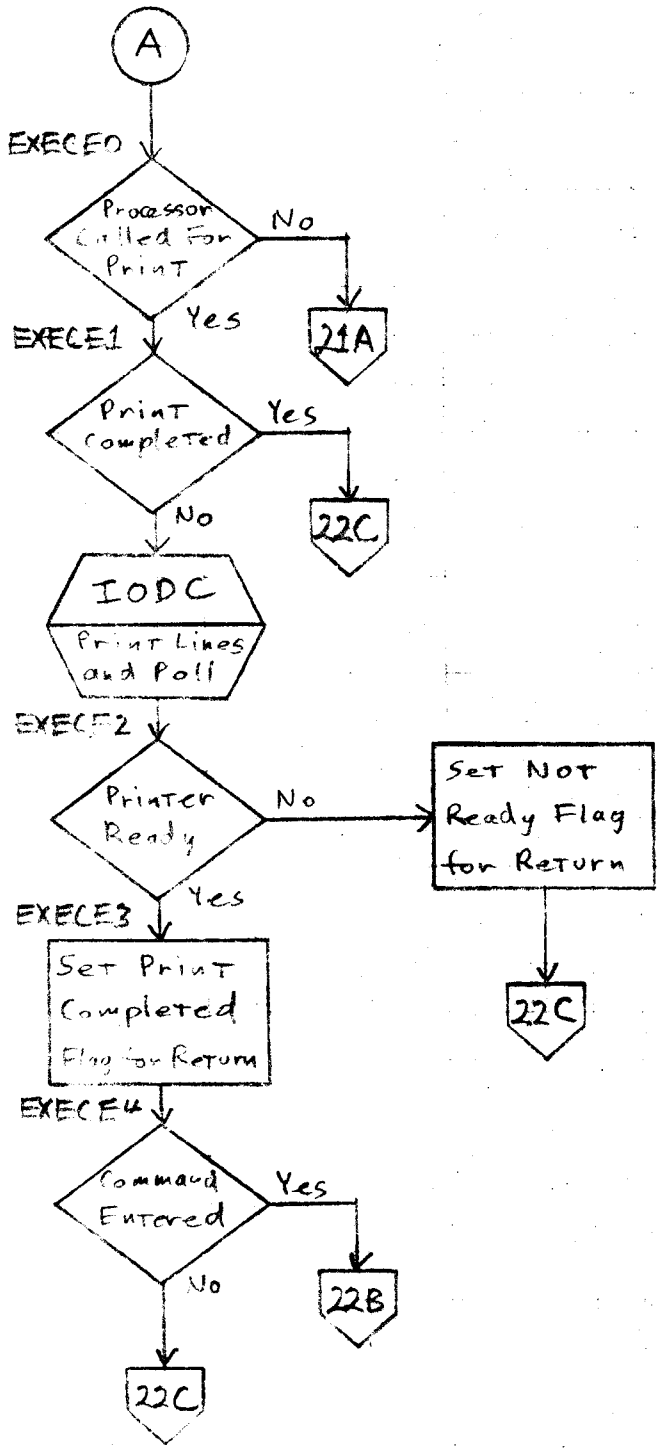

A
IØDCJ1
End of E/I200 String
No
Yes
8A
IØDCJ2
Was Storage Just Assigned
Yes
Begin another Main Line Scan
IØDC
(8)
No
IØDCJ4
Display Internally Assigned
No
Clear Display Assignment for the Subsystem
Yes
IØDCJ5
Assigned to this Subsystem
No
Request Display Assignment to the Subsystem
Yes
IØDCK0
Request Subsystem Recall
R.MTR
Drop PP
R.IDLE

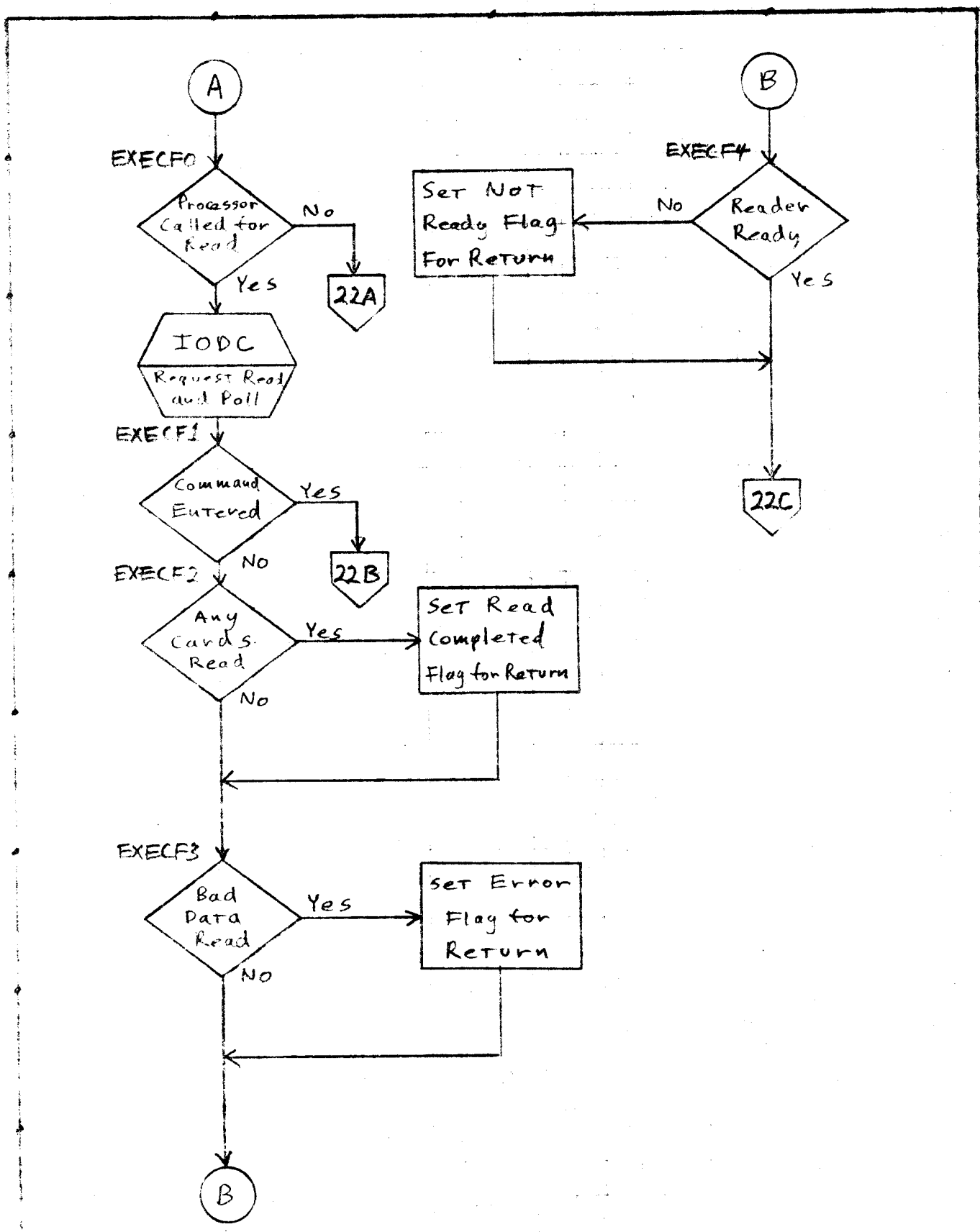

Call TO EXEC
EXEC00
Prepare Call To IODC
IODC
Perform I/O
Return TO Processor
Call To CMRL
EXEC01
Release Command
Return To Processor
Call TO WAIT
EXEC02
Initialize Call to wait
17A
Call TO IDLE
EXEC03
Initialize call to idle The Processor
17A
Call TO READ
EXEC04
Initialize call to Read Cards
17A
Call TO PRNT
EXEC05
Initialize Call to Print Lines
17A
Call To MESG
EXEC06
Initialize call to Display a Message
17A
Call TO STAL
EXEC07
Initialize call to Allocate Storage
17A

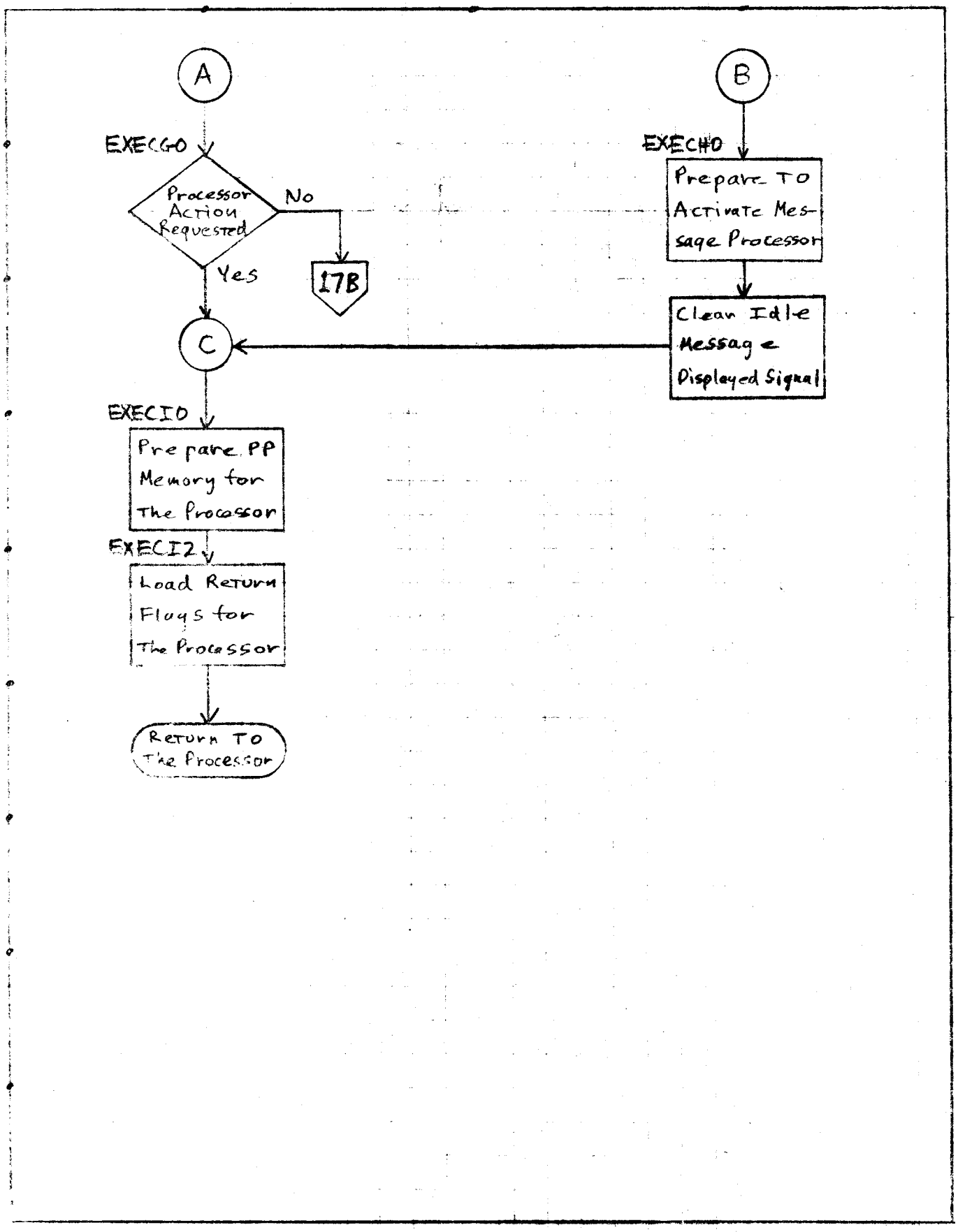

A
EXECA0
Save Call Parameters
EXECA1
Command Locked to This Processor
Yes
No
EXECA1A
Processor Called for Wait
Yes
IODC
Call For Nop
No
22C
B
EXECA2
Bring Flags For The Next Processor to the PP
EXECA3
EXECA4
No
Command Locked to any Processor
Yes
Command Locked to This Processor
No
Yes
18A
22C


A
EXECB0
End of Processor Scan
No
Yes
19A
EXECB1
All Processors Idle
Yes
No
EXECB2
Line Drop
Yes
No
EXECB3
Release all Storage and Display
IODC
Call to Drop Line
EXECC3
Clear Idle Message Displayed Signal
EXECC4
Begin New Scan of Processors
19A
EXECC0
Idle Message Displayed
No
Yes
EXECC1
Create Idle Message Signal
Message Displayed
EXECC2
IODC
Poll for 2 Seconds
IODC
Write Message to Display
Command Entered
No
Yes
A
A
22B


A
EXECD0
Processor Called For Message
No
Yes
EXECD1
Processor Called For Storage Allocate
No
Yes
20A
EXECD2
Storage Assigned
No
Yes
22C
EXECD3
Message Locked to any Processor
Yes
No
EXECD4
Message Locked to This Processor
No
Yes
17B
EXECD6
Lock Message onto This Processor
EXECD5
Message Still on The Display
No
Yes
EXECD7
IODC
Write Message to Display
EXECD8
Processor Called For storage Allocate
No
Yes
IODC
Poll for 2 Seconds
17B
EXECD9
Command Entered
Yes
No
22B
17B


A
EXEC E0
Processor Called For Print
No
21A
Yes
EXEC E1
Print Completed
Yes
22C
No
IODC
Print Lines and Poll
EXEC E2
Printer Ready
No
Set Not Ready Flag for Return
22C
Yes
EXEC E3
Set Print Completed Flag for Return
EXEC E4
Command Entered
Yes
22B
No
22C


A
EXECF0
Processor Called for Read
No
Yes
22A
IODC
Request Read and Poll
EXECF1
Command Entered
Yes
No
22B
EXECF2
Any Cards Read
Yes
No
Set Read Completed Flag for Return
EXECF3
Bad Data Read
Yes
No
Set Error Flag for Return
B
B
EXECF4
Reader Ready
No
Yes
Set Not Ready Flag For Return
22C


A
EXECGO
Processor Action Requested
No
Yes
17B
C
EXECID
Prepare PP Memory for The Processor
EXECI2
Load Return Flags for The Processor
Return TO The Processor
B
EXECHO
Prepare TO Activate Message Processor
Clear Idle Message Displayed Signal

2.4 Details of the Reader Processor (4RU)

2.4.1 Narrative of the Reader Processor Routines

PROGRAM - 4RU
ROUTINE - CRP

CRP is the main routine of the Card Reader Processor overlay and performs the tasks of 1) reading cards from the remote terminal, 2) calling CIO to write the cards to the system input device, and 3) submitting the jobs to the SCOPE input queue.

The Card Reader Processor is activated from IDLE when a READ, GO, END, CR, or LOGOUT command is to be processed. The only command recognized is READ; any other command is released and the processor returns to IDLE. When the READ command is received, STAL is called to display the message "WAITING FOR STORAGE-CR". When storage is allocated, an FET is created and the main loop is entered.

The main loop starts by calling READ to request card input data.

When control is returned to the CR Processor, either card data is available in the CM buffer, a command was entered, or some error condition was detected. If an

(2.4.1 Cont'd.)

E,CR, or LOGOUT command is received, the job currently being read is aborted (close and unload request made to CLO). The main loop is terminated and IDLE is called. Note that an E,CR, or LOGOUT received at any point in the main loop will produce this result, while a GO or READ command will continue main loop processing.

If an illegal character is detected in the data, the current job is aborted and a "CARD READ ERROR" message is sent to the terminal (via MESG) until a command is entered. If this is not an E,CR or LOGOUT, the FET parameters are re-initialized and the main loop re-entered. Note that all data currently in the buffer is lost and must be re-transmitted.

If the card reader is not ready (cards not registered and LOAD button not depressed) and no jobs have been read, or a job has been partially read (no EOF card), a "READER NOT READY" message is sent to the terminal (via MESG) until a command is entered. If a GO or READ is received the processor will continue reading. If the reader is not ready and a job has been completely read, the main loop is terminated and IDLE is called.

2.4.1 (Cont'd.)

For all card data that is received, trailing blanks are deleted and the data is moved from the IOD buffer to the CIO buffer. As each card is moved, a check is made to see if special processing is needed.

When the first card of a job is received, the transient PP program 1JU is requested. 1JU loads the system overlay 2TJ to check the job card. If an error is detected, a "JOB CARD ERROR" message is sent to the terminal via EXEC (MESG) and processing continues as in the case of an illegal character. If no errors are detected in the job card, the terminal identification is inserted into the job name which has been assigned a SCOPE sequence number by 2TJ. This new job name is written to the first word of the FET.

When an EOR or EOF is received the processor waits for any CIO operations to complete, issues a write EOR or write EOF requiest to CIO, and then waits for the write to complete. In the case of an EOF, the processor also issues a rewind request to CIO, waits for the rewind to complete and then places the job in the SCOPE input queue.

2.4.1 (Cont'd.)

The latter process consists of setting the file type to input, the control point number to zero, the disposition code to 4000_8, the last code and status to complete, and the file priority equal to the job priority. In addition the other job card parameters (CM field length, ECS field length, and time limit) are also stored in the FNT entry for SCOPE.

If there are no EOR or EOF cards to be processed but the CIO buffer contains at least one PRU of data and there is not a CIO call in progress, then a write request is issued. After the call to CIO is made (or if a call is not needed), the processor continues moving cards from the IOD buffer to the CIO buffer. If the IOD buffer becomes empty, control returns to the beginning of the main loop where another READ request is made to input new card data.

2.4.1 (Cont'd.)

PROGRAM - 4RU
SUBROUTINE - CCLO

CCLO calls CLO to close and unload the current file after waiting for any CIO call in progress to complete. It then waits for the CLO call to complete.

PROGRAM - 4RU
SUBROUTINE - CWAIT

CWAIT waits for the current CIO call, if any, to complete. If there is not a CIO call in progress, CWAIT exits immediately. Otherwise, when the call is completed, the error field in the FET is checked. If there is no error, CWAIT exits. If the error is "FNT full", the write request to CIO is re-issued. If the error is "Parity", CCLO is called and the message "INPUT FILE ERROR" is displayed at the terminal. If the command returned is E,CR or LOGOUT, then control goes to IDLE. For any other command, control goes to re-initialize the FET and read more cards.

If the error was not "FNT full" or "Parity", then the system failure procedure is followed (section 11.1) and the PP hangs until a drop or kill is issued.

2.4.1 (Cont'd.)

PROGRAM - 4RU

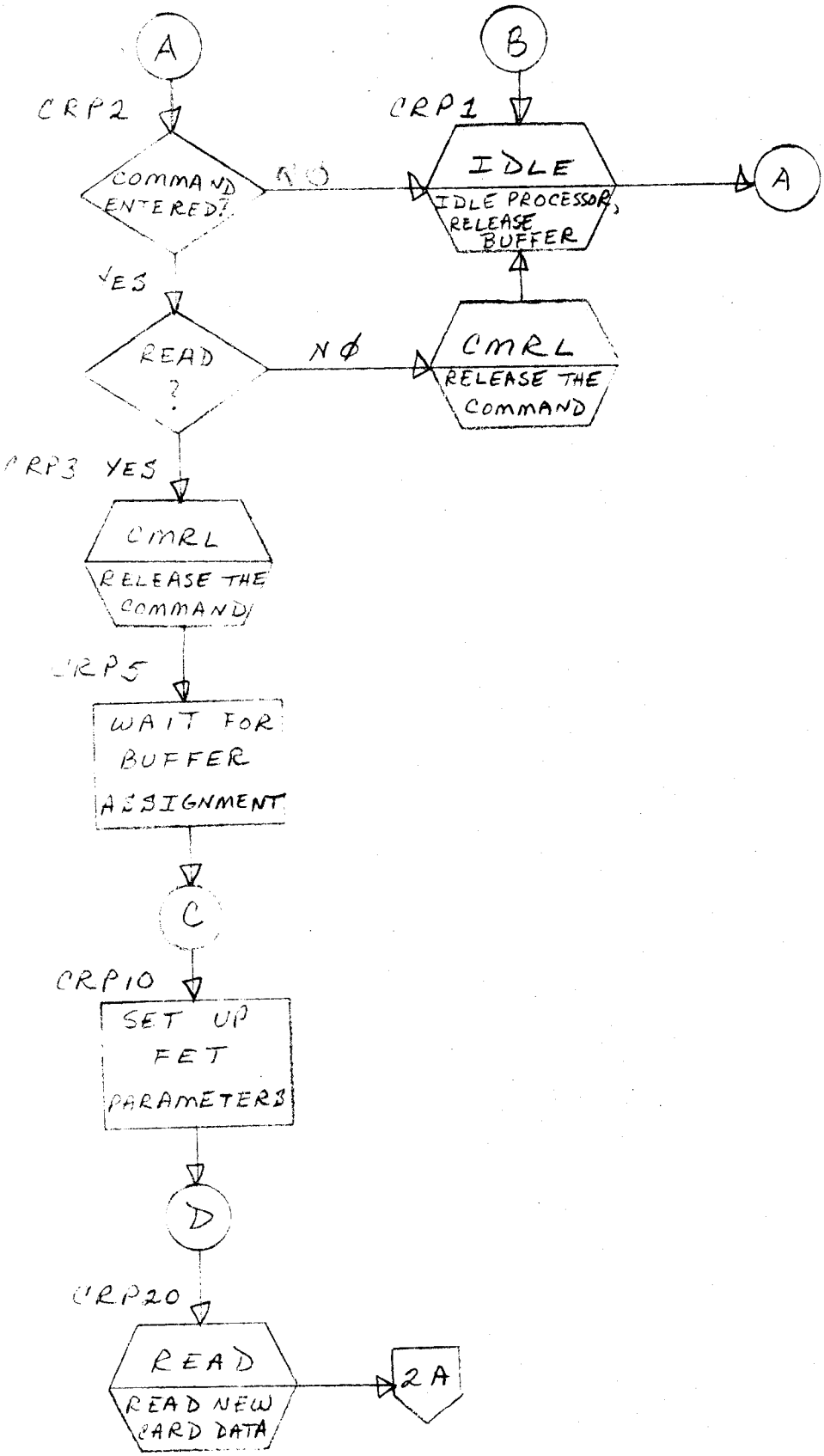
SUBROUTINE - CCIO

CCIO writes the function code for CIO into the FET status word and requests that CIO be loaded into a PP. CCIO then exits.

2.4.2 Flow Charts of the Reader Processor Routines

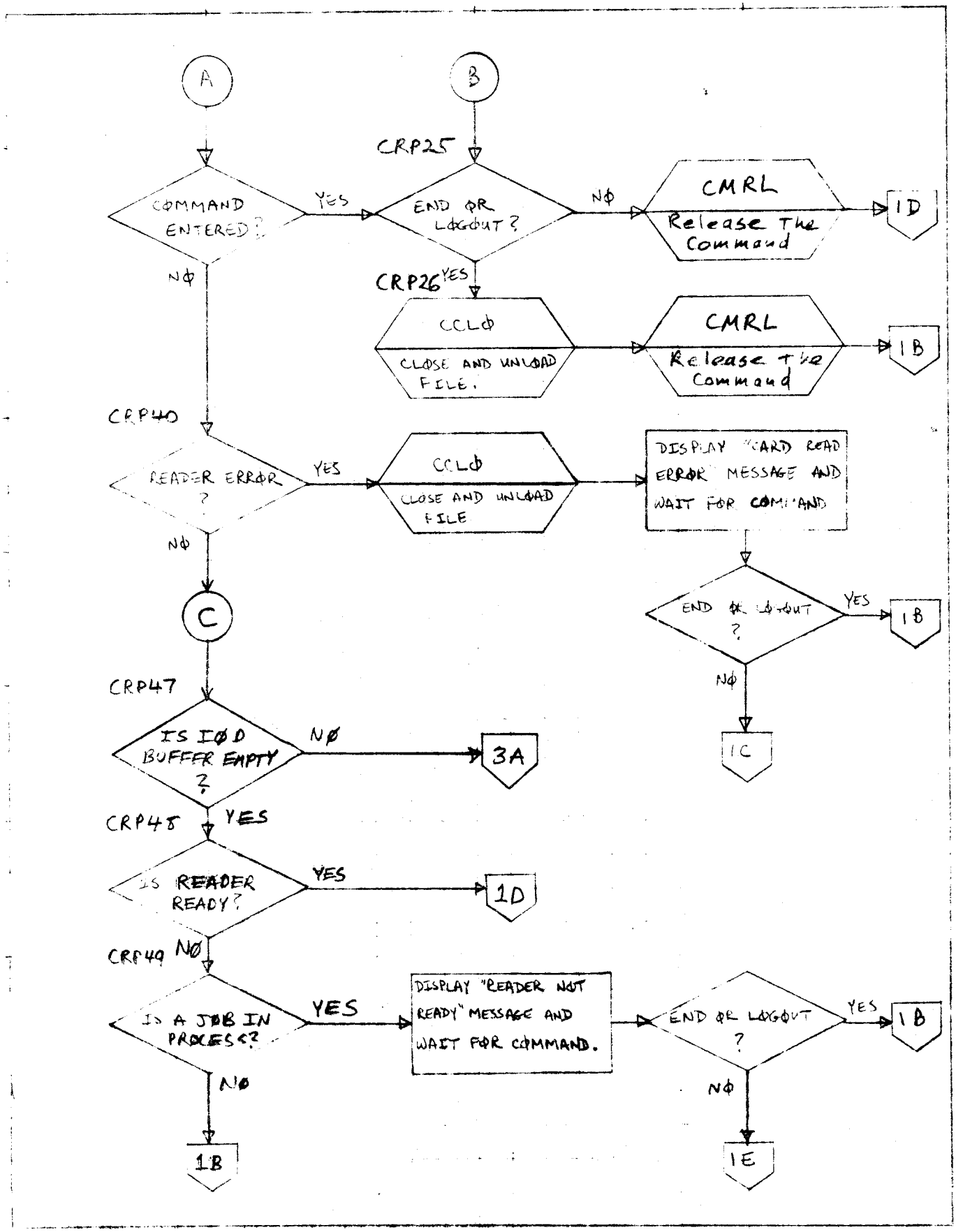
4RC

PAGE 1

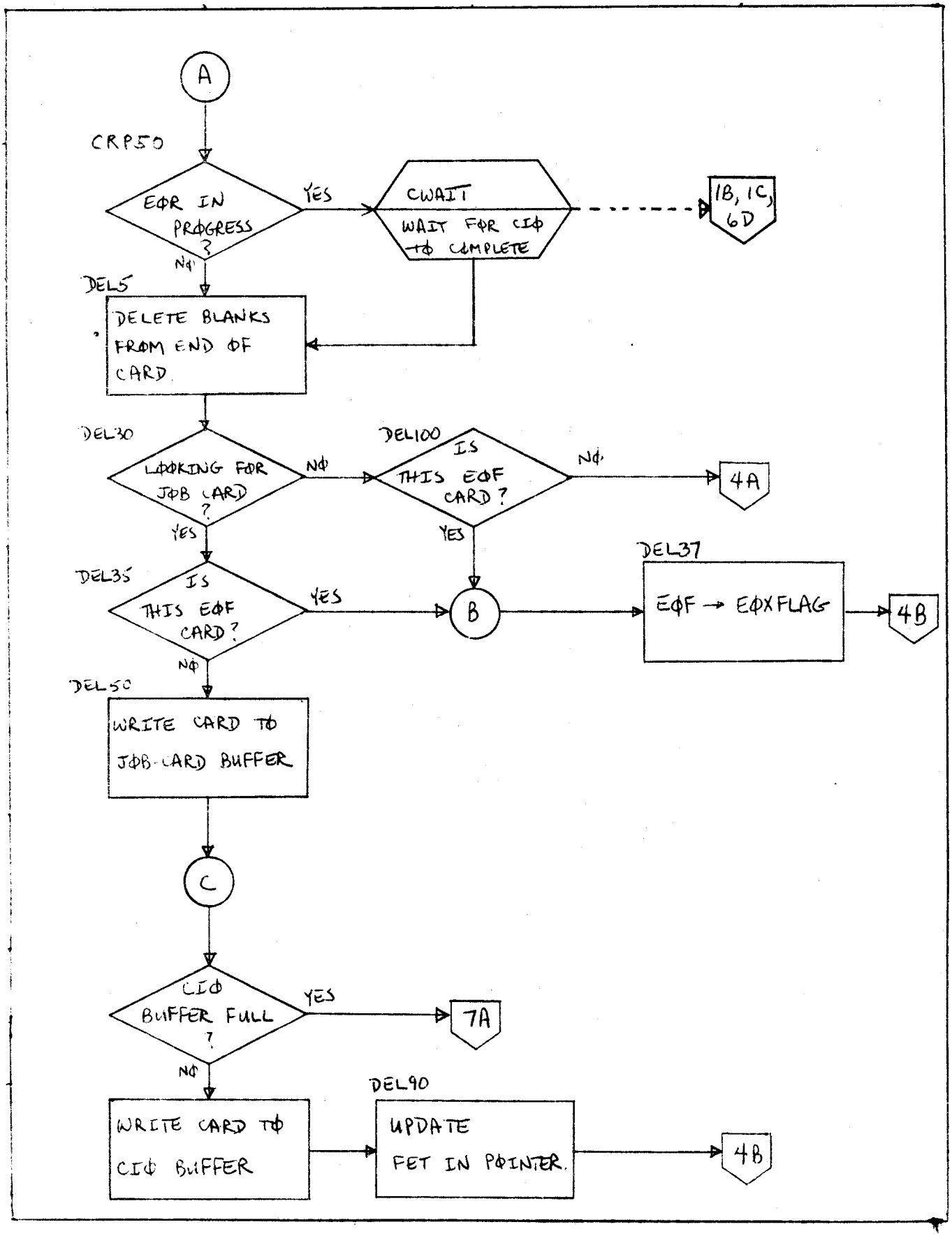

A
B
CRP2
CRP1
COMMAND ENTERED?
NO
IDLE
IDLE PROCESSOR, RELEASE BUFFER
A
YES
READ ?
NO
CMRL
RELEASE THE COMMAND
CRP3 YES
CMRL
RELEASE THE COMMAND
CRP5
WAIT FOR BUFFER ASSIGNMENT
C
CRP10
SET UP FET PARAMETERS
D
CRP20
READ
READ NEW CARD DATA
2A

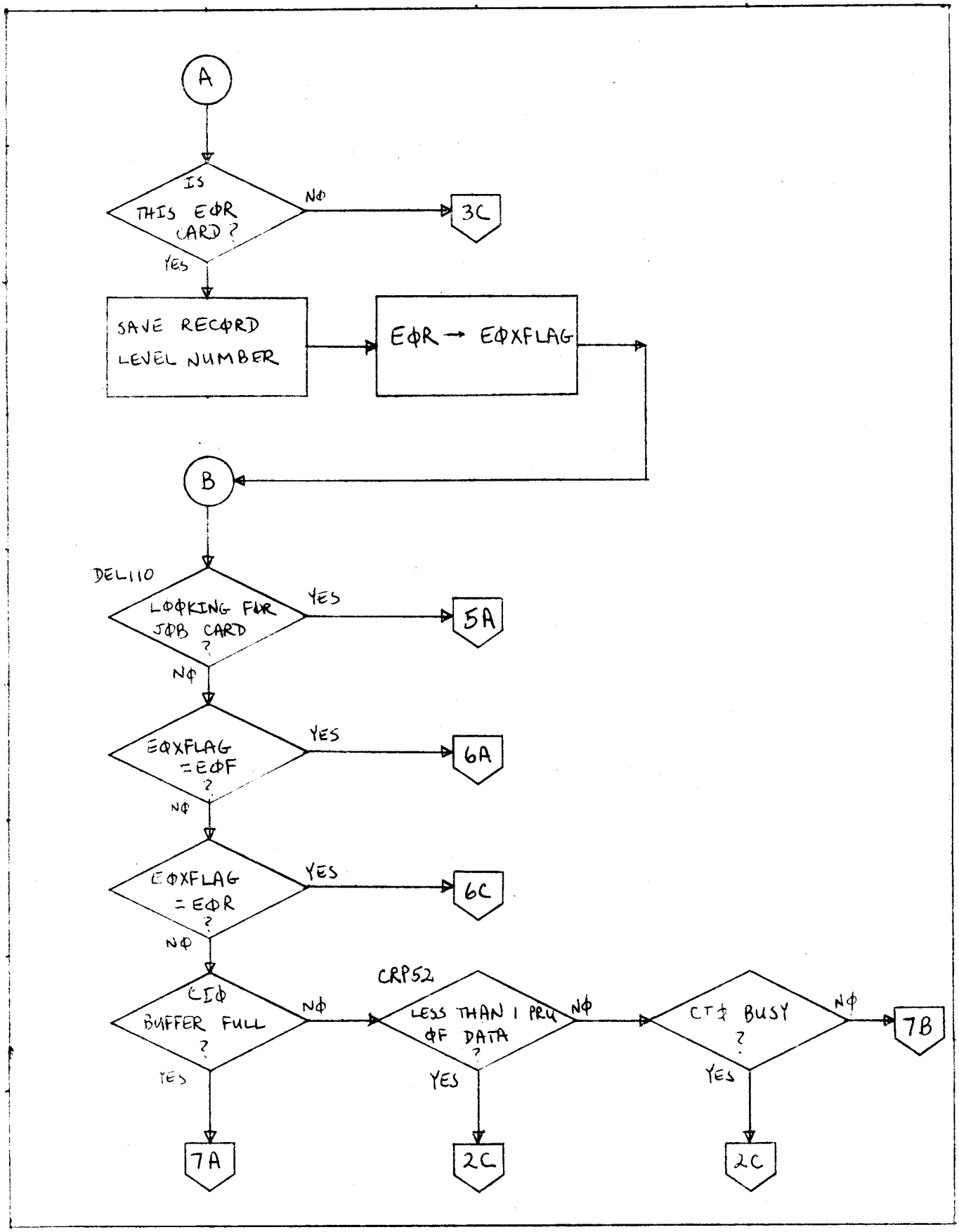
4RU

Page 2

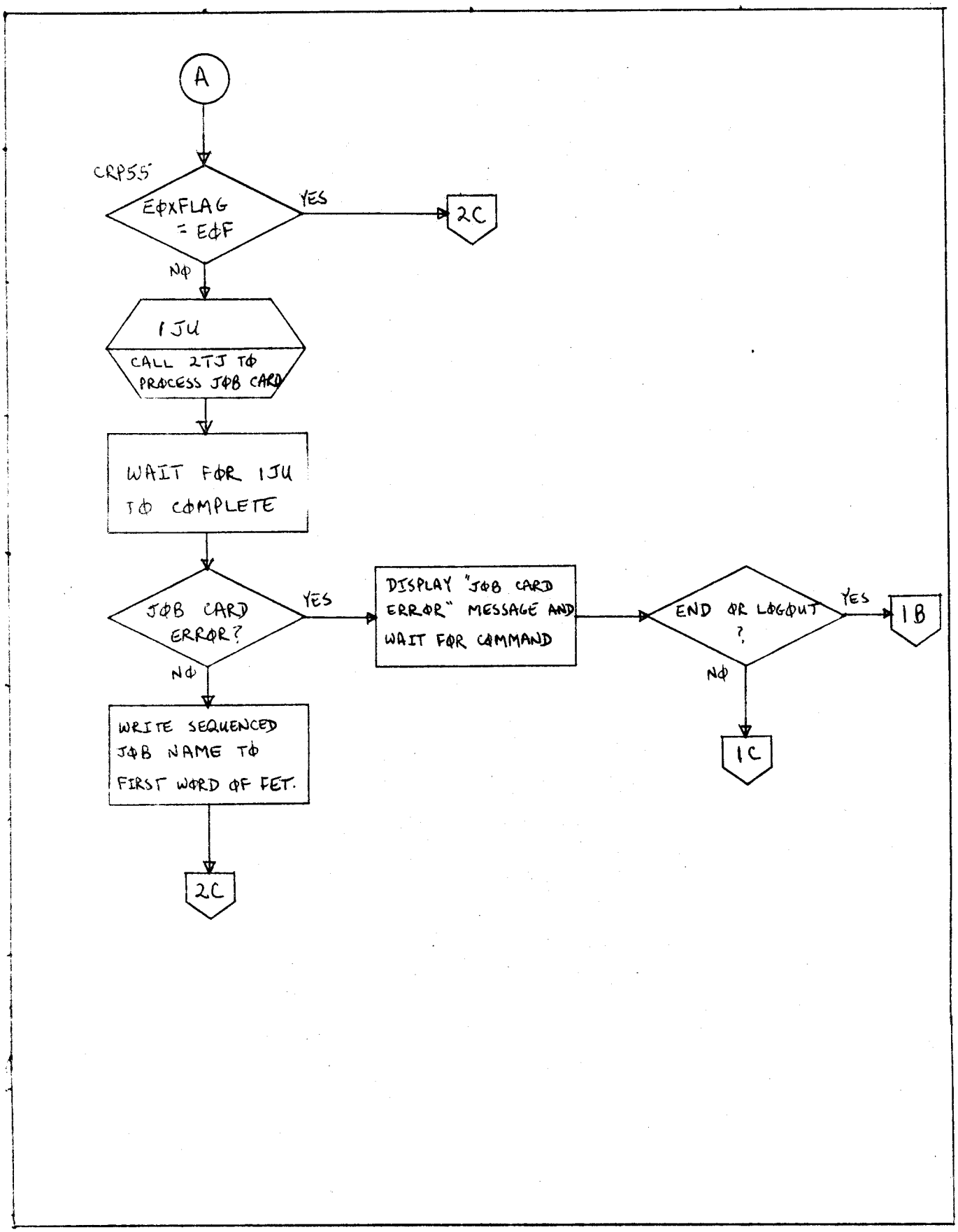

A
B
CRP25
COMMAND ENTERED?
YES
END OR LOGOUT?
NØ
CMRL
Release The Command
1D
NØ
CRP26
YES
CCLØ
CLOSE AND UNLOAD FILE.
CMRL
Release The Command
1B
CRP40
READER ERROR?
YES
CCLØ
CLOSE AND UNLOAD FILE
DISPLAY "CARD READ ERROR" MESSAGE AND WAIT FOR COMMAND
END OR LOGOUT?
YES
1B
NØ
1C
NØ
C
CRP47
IS IØD BUFFER EMPTY?
NØ
3A
CRP48
YES
IS READER READY?
YES
1D
CRP49
NØ
IS A JOB IN PROCESS?
YES
DISPLAY "READER NOT READY" MESSAGE AND WAIT FOR COMMAND.
END OR LOGOUT?
YES
1B
NØ
NØ
1B
1E

PAGE 3

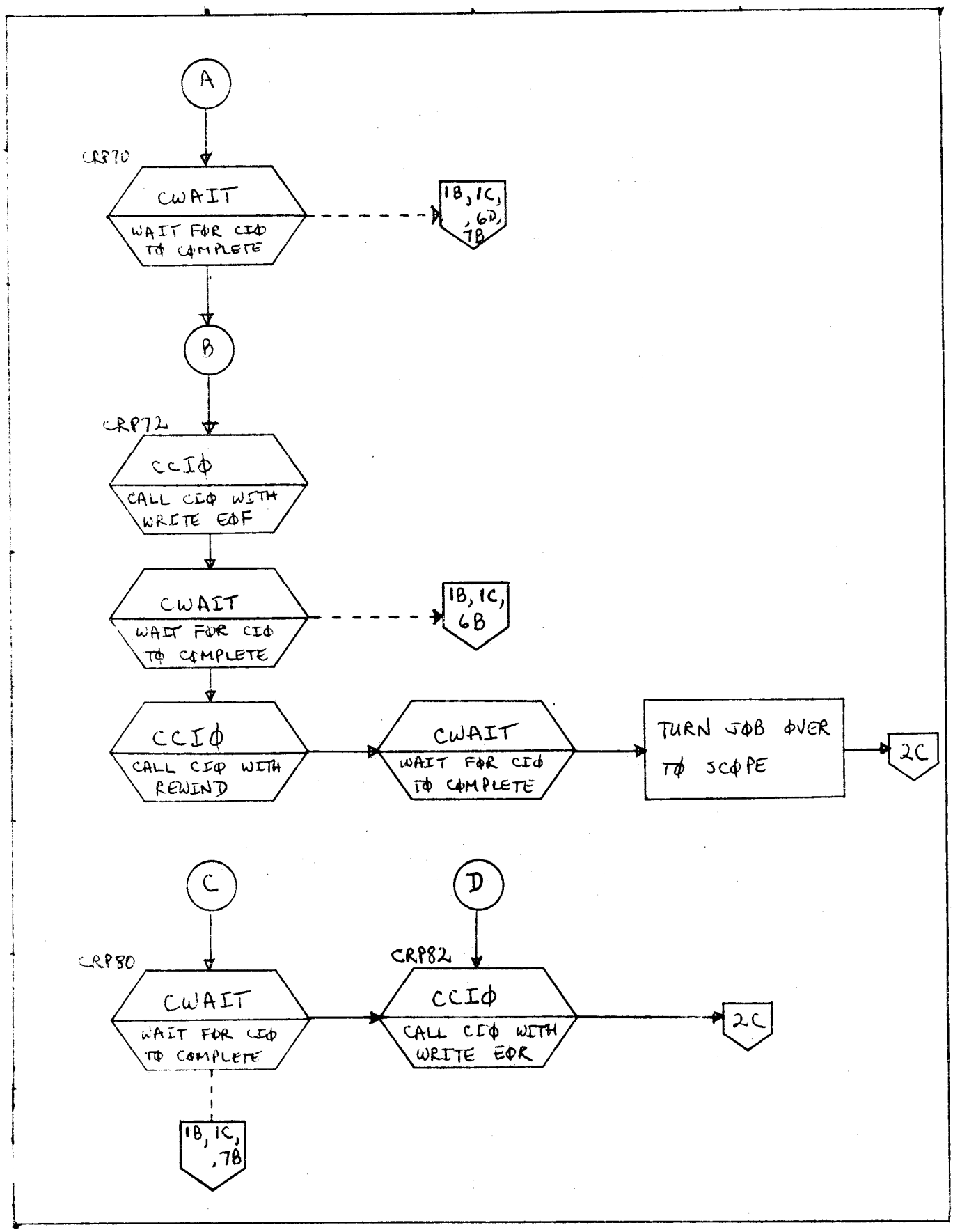

A
CRP50
EOR IN PROGRESS ?
YES
CWAIT
WAIT FOR CIO TO COMPLETE
1B, 1C, 6D
NO
DEL5
DELETE BLANKS FROM END OF CARD
DEL30
LOOKING FOR JOB CARD ?
NO
DEL100
IS THIS EOF CARD ?
NO
4A
YES
YES
DEL35
IS THIS EOF CARD ?
YES
B
DEL37
EOF → EOXFLAG
4B
NO
DEL50
WRITE CARD TO JOB CARD BUFFER
C
CIO BUFFER FULL ?
YES
7A
NO
DEL90
WRITE CARD TO CIO BUFFER
UPDATE FET IN POINTER.
4B

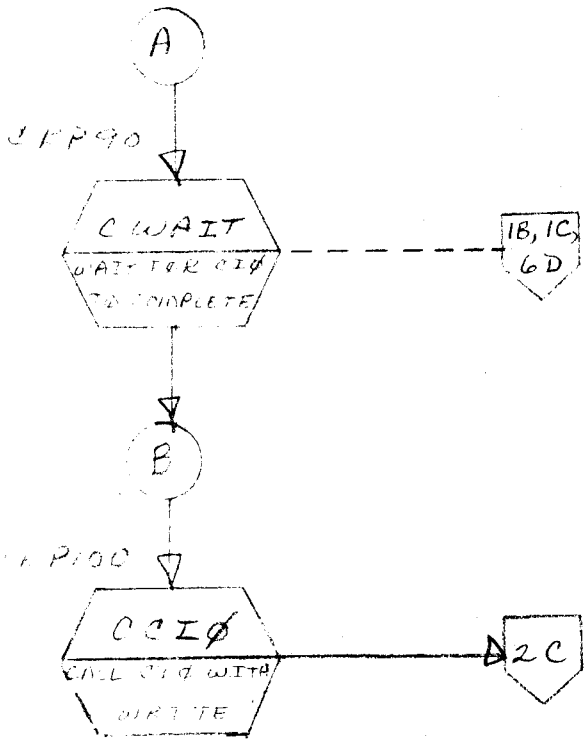

A
IS THIS EOR CARD ?
NO
3C
YES
SAVE RECORD LEVEL NUMBER
EOR → EOXFLAG
B
DEL110
LOOKING FOR JOB CARD ?
YES
5A
NO
EOXFLAG = EOF ?
YES
6A
NO
EOXFLAG = EOR ?
YES
6C
NO
CI0 BUFFER FULL ?
NO
CRP52
LESS THAN 1 PRU OF DATA ?
NO
CT≠ BUSY ?
NO
7B
YES
7A
YES
2C
YES
2C

PAGE 5

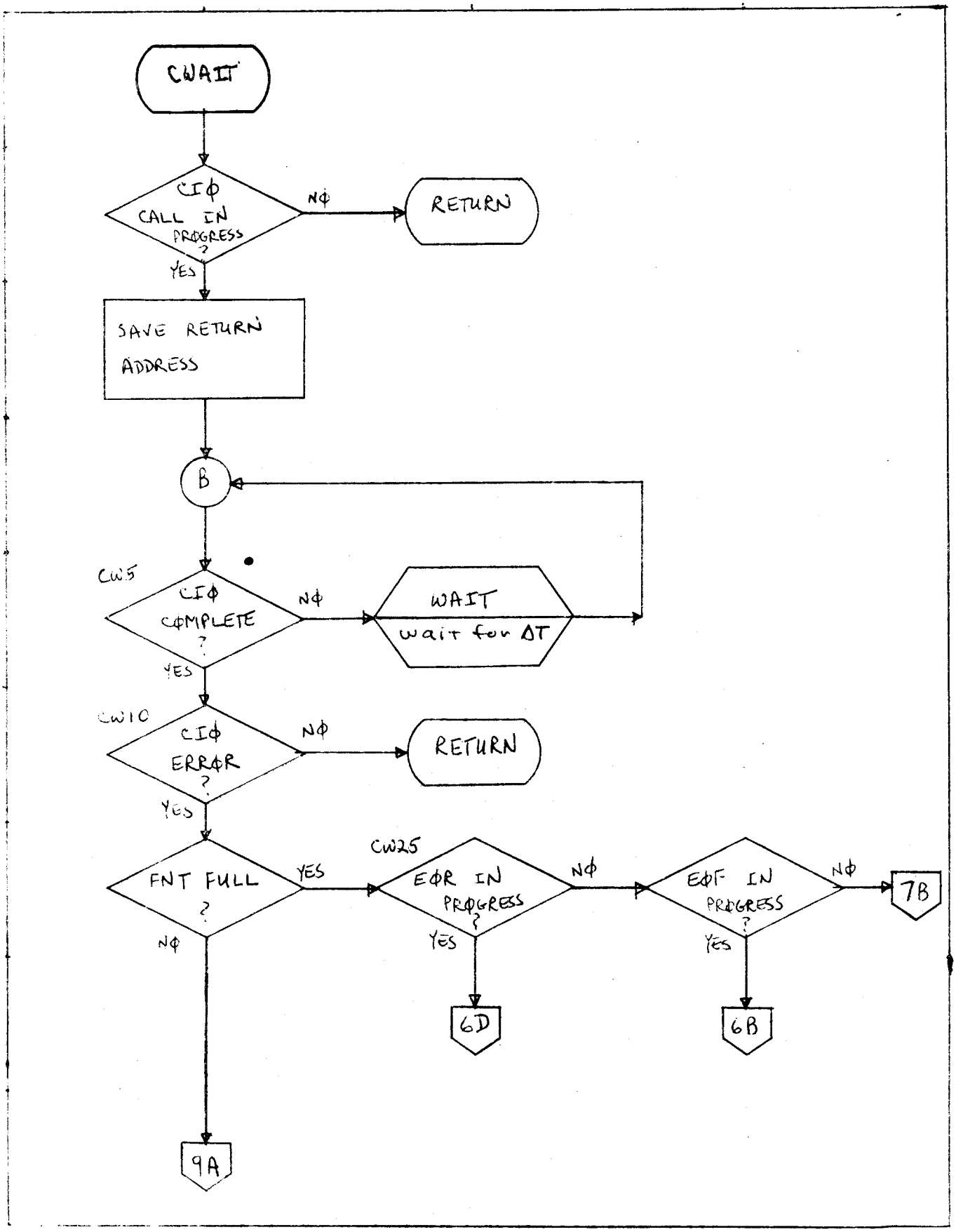

A
CRP55
EOXFLAG = EOF
YES
2C
NO
1JU
CALL 2TJ TO PROCESS JOB CARD
WAIT FOR 1JU TO COMPLETE
JOB CARD ERROR?
YES
DISPLAY "JOB CARD ERROR" MESSAGE AND WAIT FOR COMMAND
END OR LOGOUT ?
YES
1B
NO
1C
NO
WRITE SEQUENCED JOB NAME TO FIRST WORD OF FET.
2C

PAGE 6

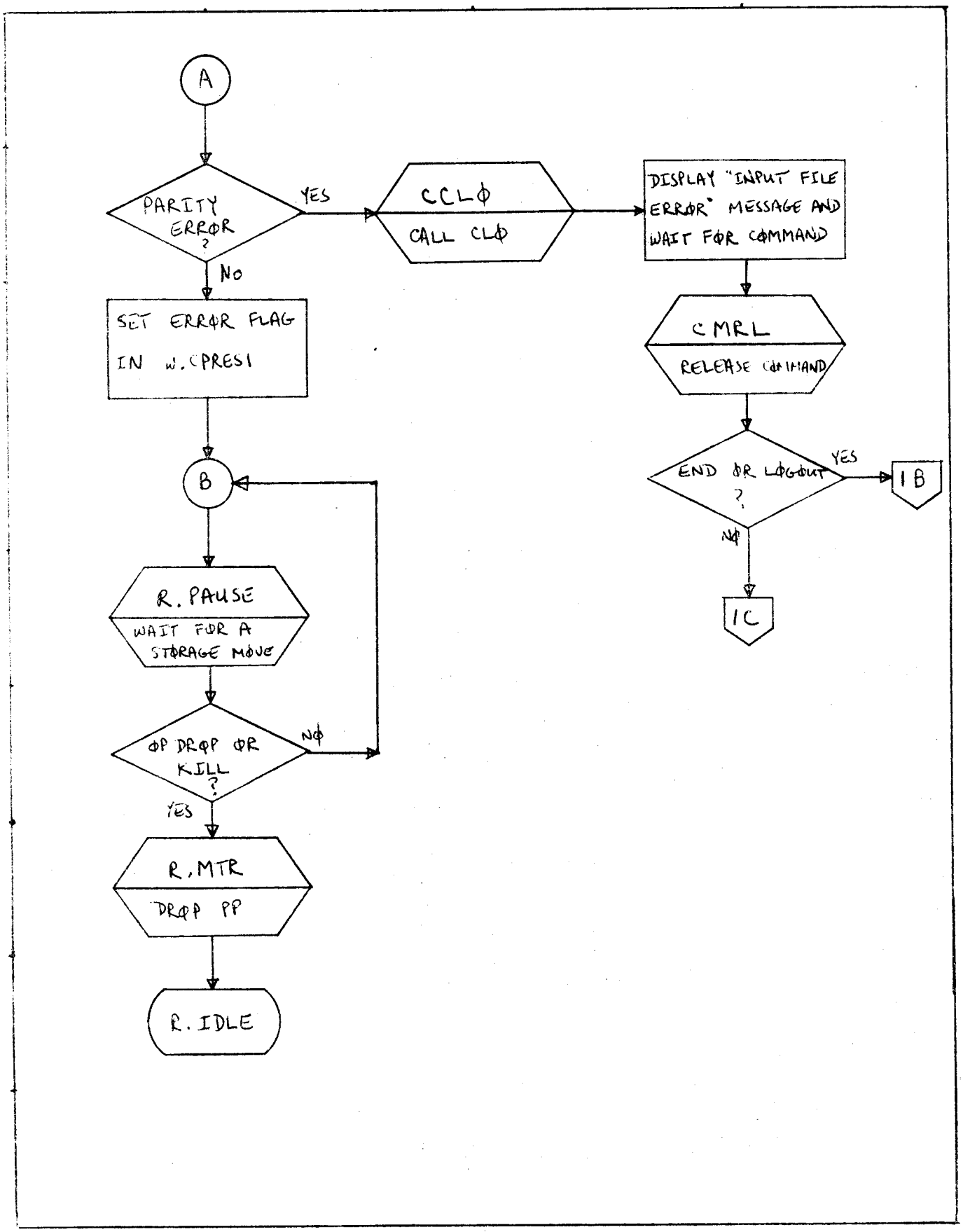

A
CRP70
CWAIT
WAIT FOR CIO TO COMPLETE
1B, 1C, 6D, 7B
B
CRP72
CCIO
CALL CIO WITH WRITE EOF
CWAIT
WAIT FOR CIO TO COMPLETE
1B, 1C, 6B
CCIO
CALL CIO WITH REWIND
CWAIT
WAIT FOR CIO TO COMPLETE
TURN JOB OVER TO SCOPE
2C
C
D
CRP80
CRP82
CWAIT
WAIT FOR CIO TO COMPLETE
CCIO
CALL CIO WITH WRITE EOR
2C
1B, 1C, 7B

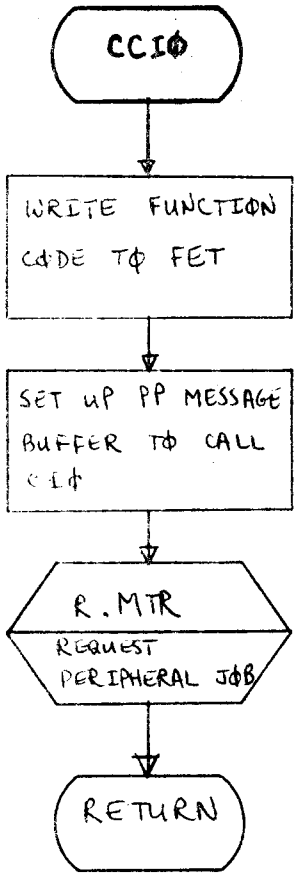

A
CWAIT
WAIT FOR CIØ TO COMPLETE
1B, 1C, 6D
B
CCIØ
CALL CIØ WITH WRITE
2C

PAGE 8

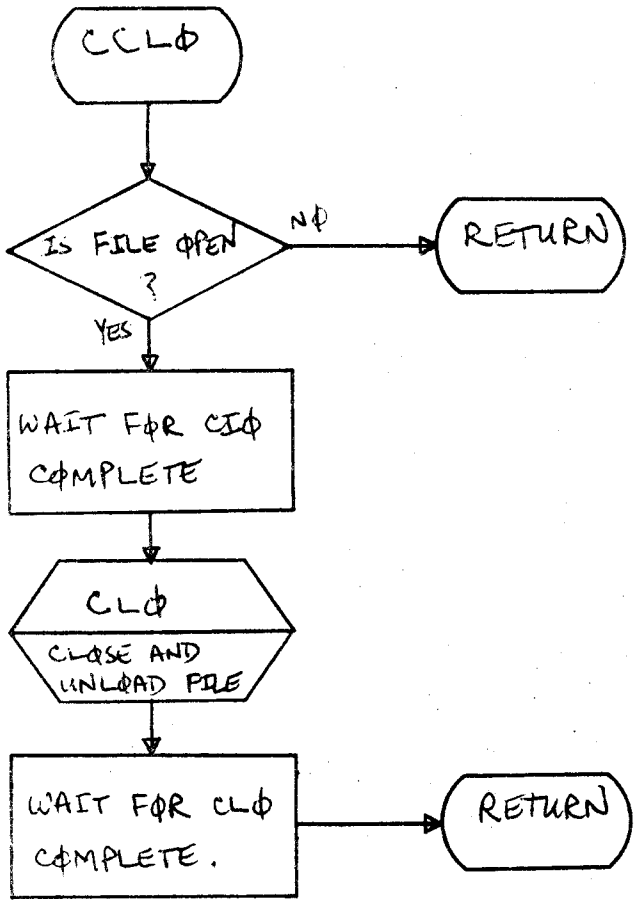

CWAIT
CIΦ CALL IN PROGRESS ?
NΦ
RETURN
YES
SAVE RETURN ADDRESS
B
CW5
CIΦ COMPLETE ?
NΦ
WAIT
wait for ΔT
YES
CW10
CIΦ ERRΦR ?
NΦ
RETURN
YES
FNT FULL ?
YES
CW25
EΦR IN PROGRESS ?
NΦ
EΦF IN PROGRESS ?
NΦ
7B
NΦ
YES
YES
6D
6B
9A

PAGE 9


A
PARITY ERROR ?
YES
CCLϕ
CALL CLϕ
DISPLAY "INPUT FILE ERRϕR" MESSAGE AND WAIT FϕR CϕMMAND
No
SET ERRϕR FLAG IN W.CPRES1
CMRL
RELEASE CϕMMAND
END ϕR LϕGϕUT ?
YES
1B
Nϕ
1C
B
R.PAUSE
WAIT FϕR A STϕRAGE MϕVE
ϕP DRϕP ϕR KILL ?
Nϕ
YES
R.MTR
DRϕP PP
R.IDLE


CCIØ
WRITE FUNCTION CODE TØ FET
SET UP PP MESSAGE BUFFER TØ CALL CIØ
R.MTR
REQUEST PERIPHERAL JØB
RETURN

PAGE 11


CCLØ
IS FILE ØPEN ?
NØ
RETURN
YES
WAIT FØR CIØ CØMPLETE
CLØ
CLØSE AND UNLØAD FILE
WAIT FØR CLØ CØMPLETE.
RETURN

2.4.3 Reader Processor Save Area and CM Buffer

Save Area: SAV1-SAV5, direct cells 63B-67B

SAV1 = CIOW

SAV2 = RDRESP

SAV3 = CLOSER

SAV4 = COUNT

SAV5 = CWAITR

CIOW

C2TJ - This bit is set when the next card is to be checked by 2TJ.

MFOP - This bit is set when the input file is open.

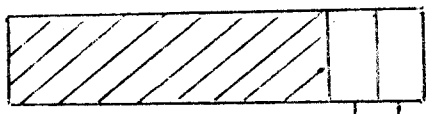
WRINP - This bit is set when a CIO request is in process.

EORINP- This bit is set when a CIO EOR write request is in process.

EOFINP- This bit is set when a CIO EOF write request is in process.

SRDNOT- This bit is set when a "Reader Not Ready" message is to be sent to the terminal if a not ready condition is detected.

RDRESP



This bit is set when a bad card code is detected.

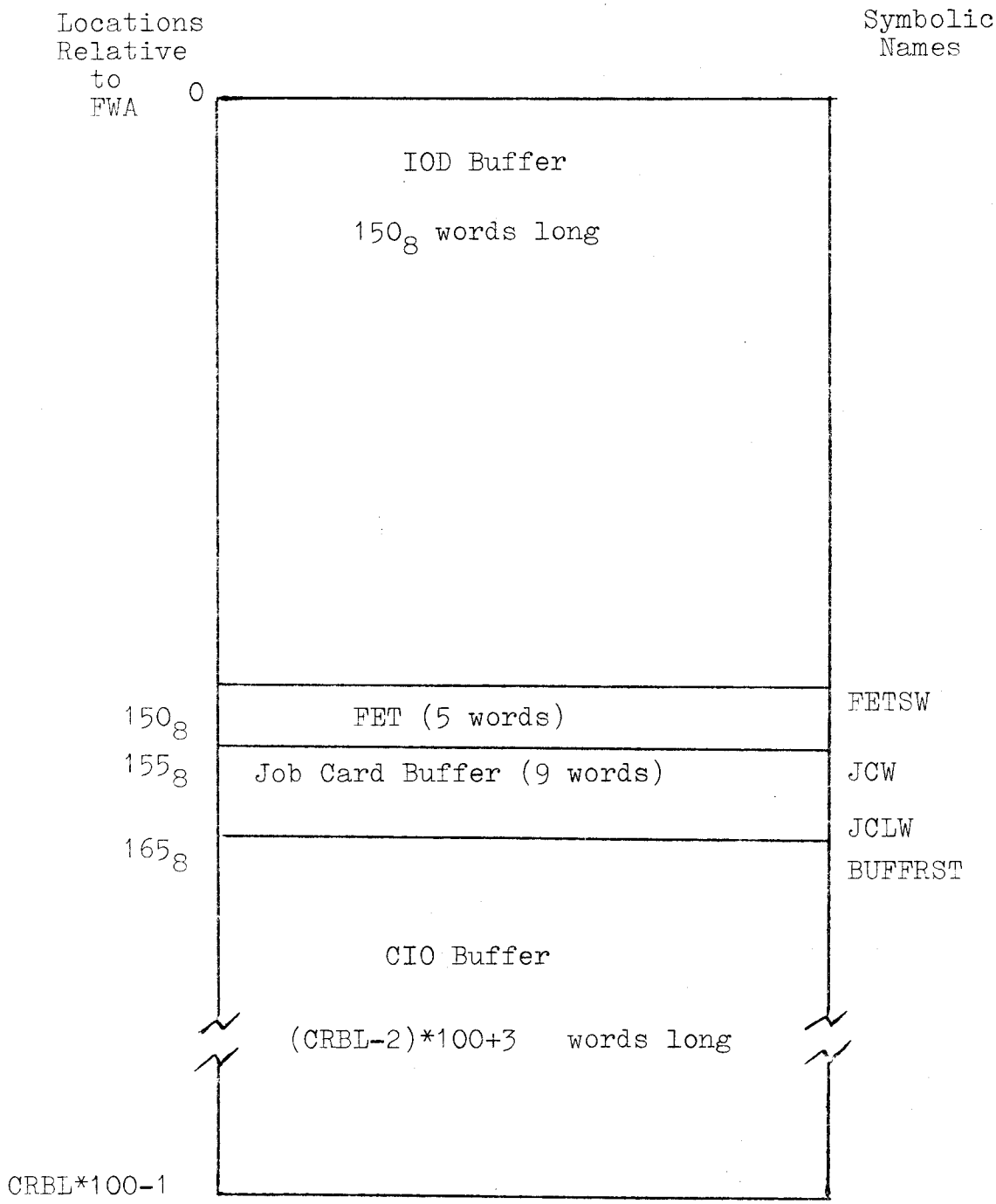
This bit is set when a reader not ready condition is detected.

CLOSER is the return address for CLOSE.

COUNT is the number of words in the IOD buffer which have already been processed.

CWAITR is the return address for CWAIT.

CM Buffer: first word address = FWA=BAD*100_8+RA
CRBL*100_8 words long ($\geq 300_8$)



2.5 Details of the Print Processor (4PU)

2.5.1 Narrative of the Print Processor Routines

PROGRAM - 4PU

ROUTINE - PRTP

The print processor is activated from IDLE when FNTS finds an output file to be printed, or a command for the print processor is entered. A command (if any) is processed and then printing is initialized if a print file was found. Initialization consists of testing for the printer ready, requesting a buffer, setting up an FET for the file and printing the banner. The banner is generated by the print processor from the file name, and is printed twice.

After initialization the print processing main loop is entered. The functions of the main loop are as follows:

1) Read output data from the CIO buffer, perform blank and zero compression, detect PM messages, and then write the modified data to the IOD buffer.
2) Modify the CIO buffer pointers, check for more data needed, make requests to CIO, and check the status of previous CIO requests.

3) Call PRNT to print output data, MESG to display PM messages, or WAIT to wait for CIO to complete.

Examination and modification of the output data occurs first, starting at the OUT address of the CIO buffer, and is terminated when

1) the total number of characters to be transmitted exceeds 1000, or
2) the CIO buffer is empty, or
3) a PM message is detected.

If a PM message is the first line encountered when examination starts, the message is transferred to the IOD buffer and OUT is reset to the beginning of the next line. After the resulting command is processed, the main loop is restarted to examine the data following the PM message.

If a print line is the first line encountered when examination starts, then blank and zero compression is performed as the data is transferred to the IOD buffer.

When examination terminates, OUT is reset to the beginning of the first non-printable line (a PM message or a partial line).

If no complete lines were written into the buffer for IOD and an end-of-information was detected on the last completed CIO read, the file is closed and the processor deactivates by calling IDLE.

After OUT is reset, a CIO call is attempted if there are 65 or more empty words in the CIO buffer. If CIO is busy or if the buffer is too full, the call attempt is bypassed. Otherwise, the previous call

is checked for errors and a new call made. If an end-of-information is detected, no new calls are made. If a parity error occurs, MESG is called to display the message "OUTPUT FILE ERROR". After the resulting command is processed, the CIO call is retried.

After the CIO call attempt and if data had been transferred to the IOD buffer, then PRNT is called to print the data. Otherwise, WAIT is called. The main loop is restarted after printing or waiting is complete.

If, after any print operation, the printer is found to be not ready, MESG is called to display the message "PRINTER-NOT-READY". After the resulting command is processed, the print operation is retried.

PROGRAM - 4PU SUBROUTINE - COMP

COMP is the main command processing subroutine. A return is made after the command processing is complete with the following exceptions:

1) A SUSPEND command will prevent return until a CONTINUE is entered, and
2) Any command which drops the output file (LOGOUT, OUTPUT, or two successive E,LP commands) will cause COMP to deactivate the print processor after the command is executed.

COMP will always wait until any required file I/O is complete and will reset the FET appropriately before returning.

PROGRAM - 4PU

SUBROUTINES - CCIO, CCLO

These are used to format and initiate calls to CIO and CLO, respectively.

PROGRAM - 4PU

SUBROUTINE - CWAIT

CWAIT waits until a CIO or CLO call is complete. If an error is flagged that is not EOI or a parity error, the system failure procedure is followed (Section 11.1) and the PP hangs until a drop or kill is issued.

PROGRAM - 4PU

SUBROUTINES - ACHS, DCHS

These are used to assign and release the FNT and FST channels.

PROGRAM - 4PU

SUBROUTINES - RELCVT, ABSCVT

These are used to convert between addresses relative to RA, and addresses relative to the buffer area.

PROGRAM - 4PU

SUBROUTINES - SUBUFPT, RDBUFPT

These are used to write and read the FET.

PROGRAM - 4PU

SUBROUTINE - EXERF

EXERF is used to extract the error field from the FET code and status word.

PROGRAM - 4PU

SUBROUTINE - SETOUT

SETOUT is used to reset the value of OUT in the FET.

PROGRAM - 4PU

SUBROUTINE - WWRD

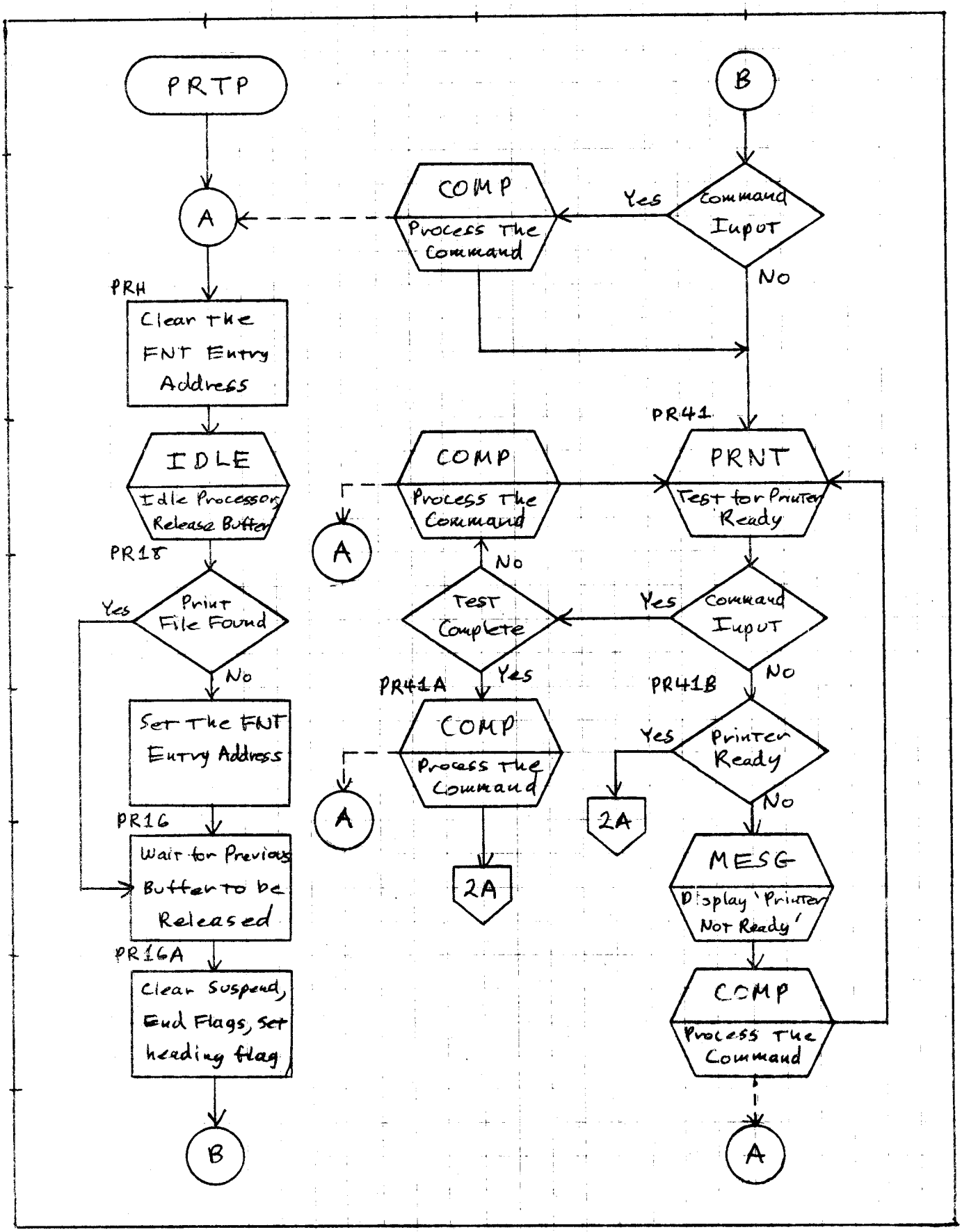
WWRD is used during banner generation to write words to the IOD buffer.

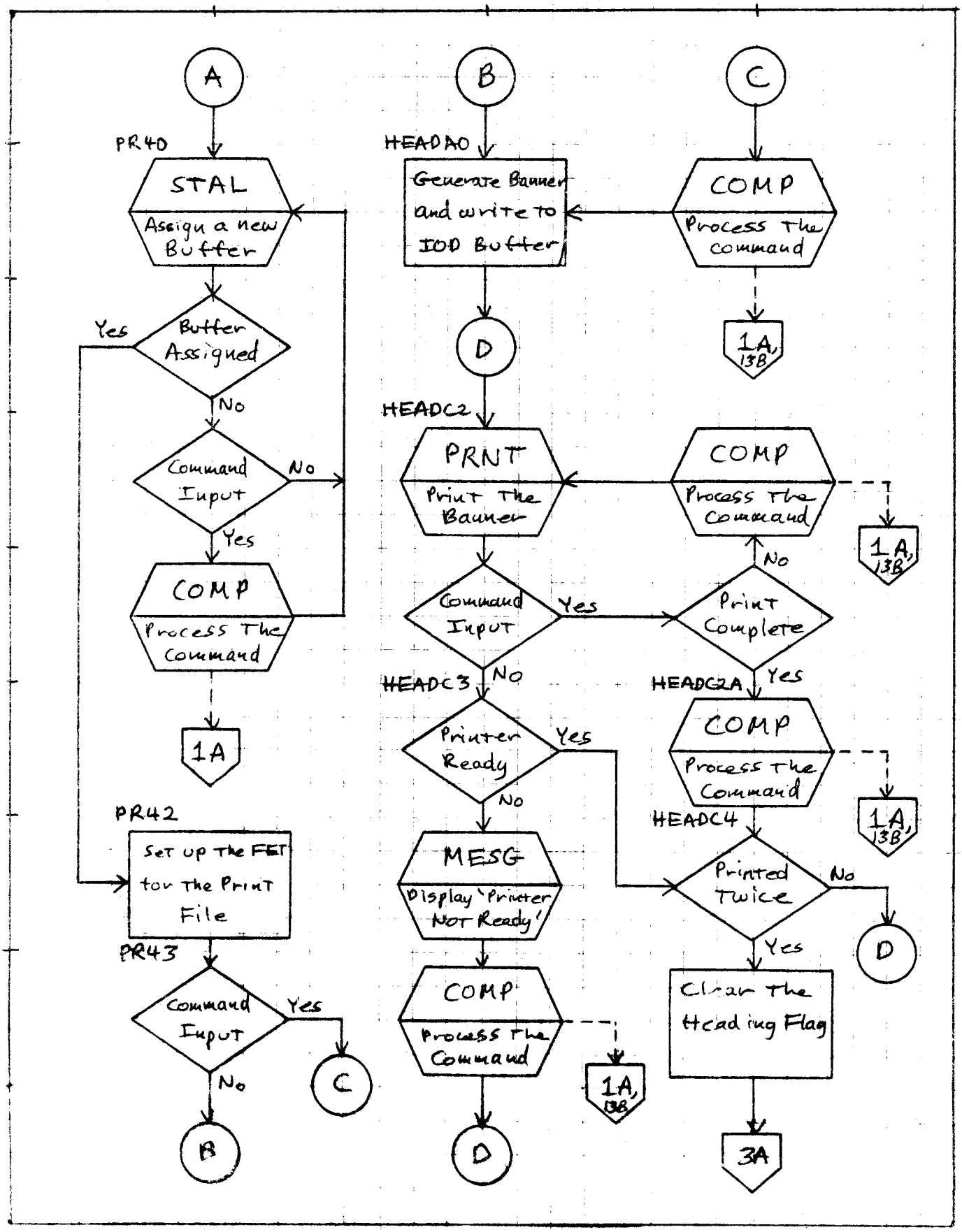
2.5.2 Handling of Non-standard Print Files

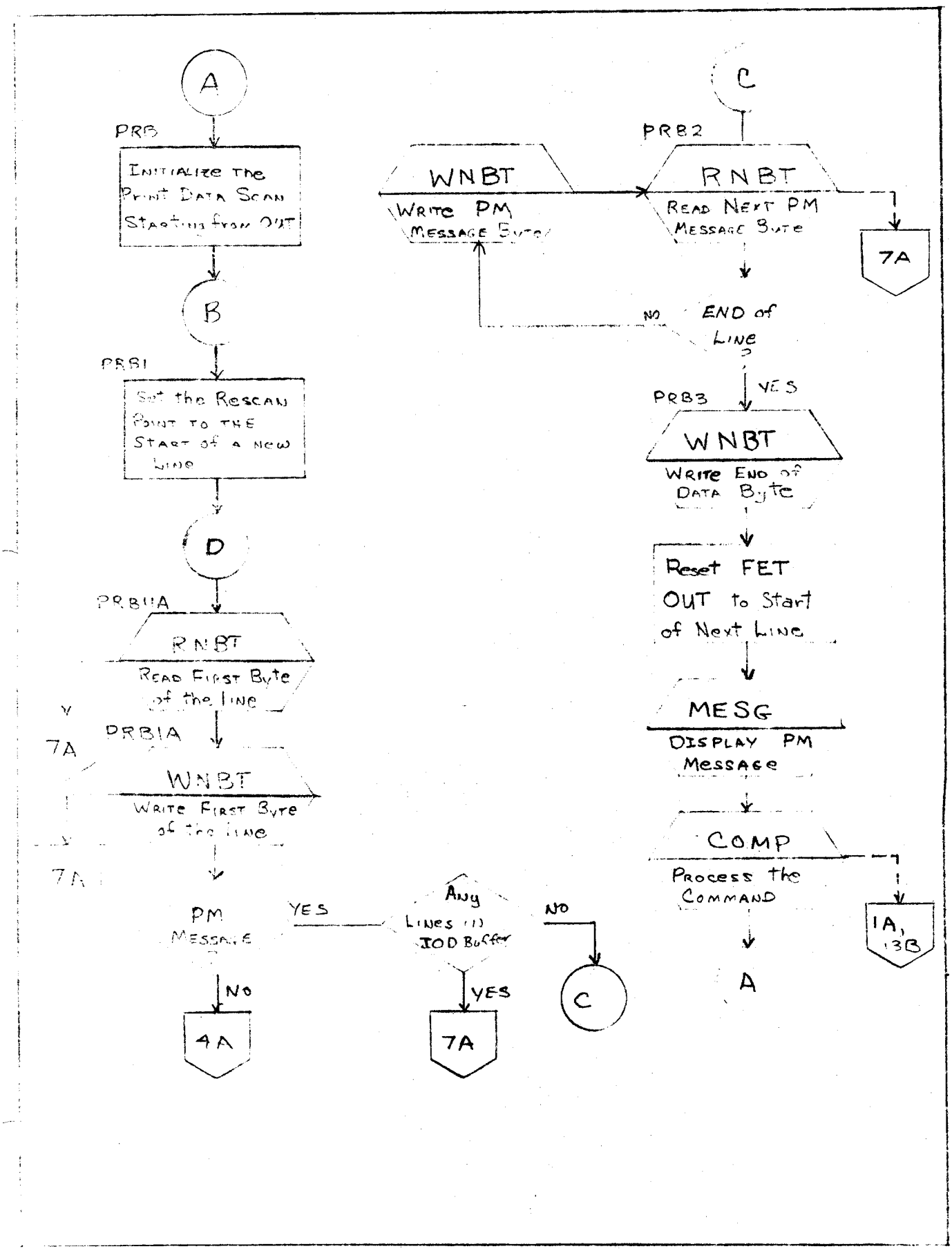
The print processor considers a standard line to be a string of 69 or less 12 bit data bytes terminated by a 12 bit zero byte. The first byte of the string starts in the high order byte of the first word. Each data byte consists of two non-zero 6 bit display codes. A zero byte occurring in any byte of a central memory word other than byte four will be replaced with two blank characters. A zero byte occurring in byte 4 of a central memory word (last byte) is treated as End-of-Line. Any data byte containing a zero display code (00XX or XX00) will have the zero code printed as a blank.

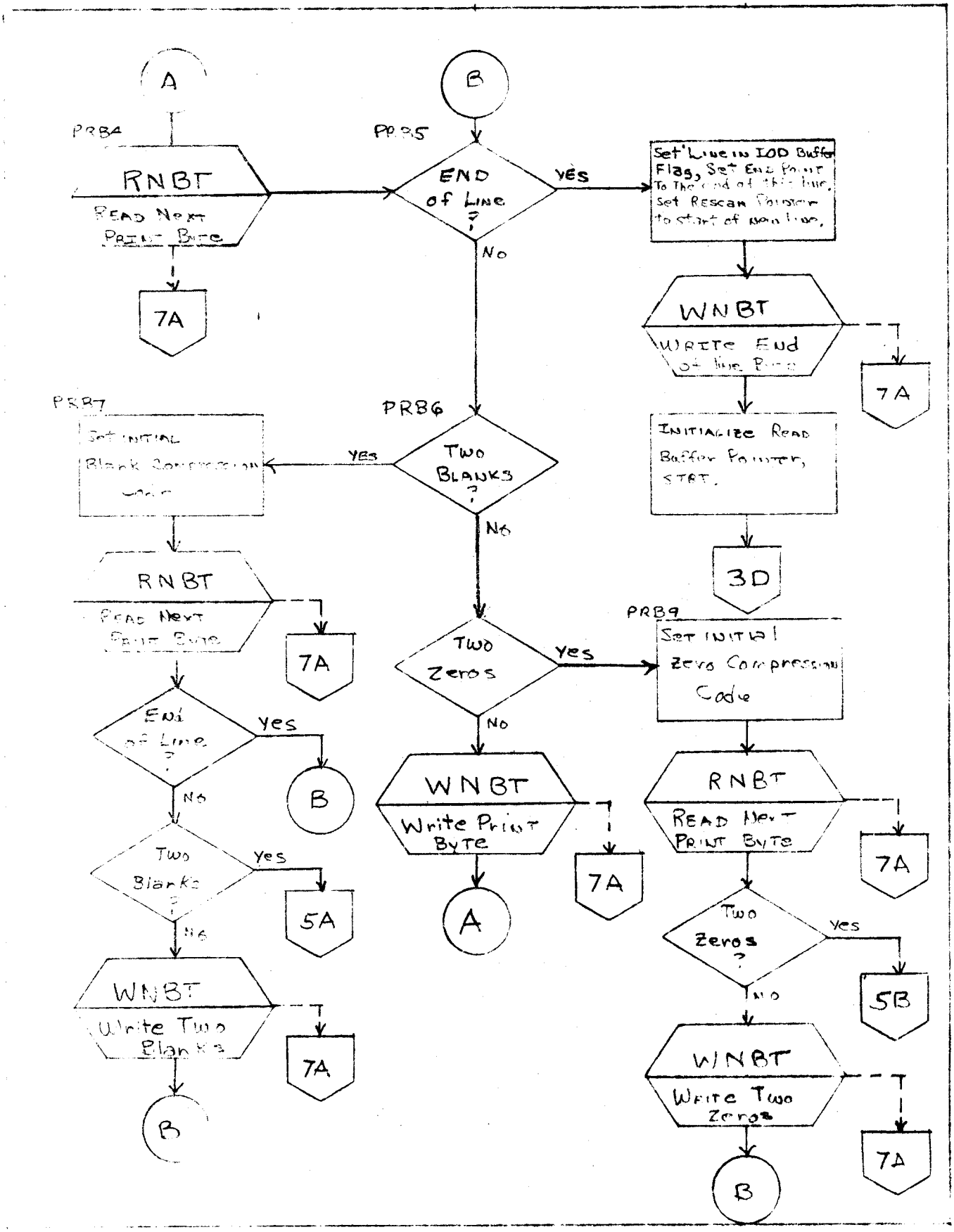
Non-standard files with lines greater than 69 bytes (carriage control + 136 printing characters) will be truncated to 69 bytes and printed. The remaining data up to the end-of-line (zero byte) will be ignored.

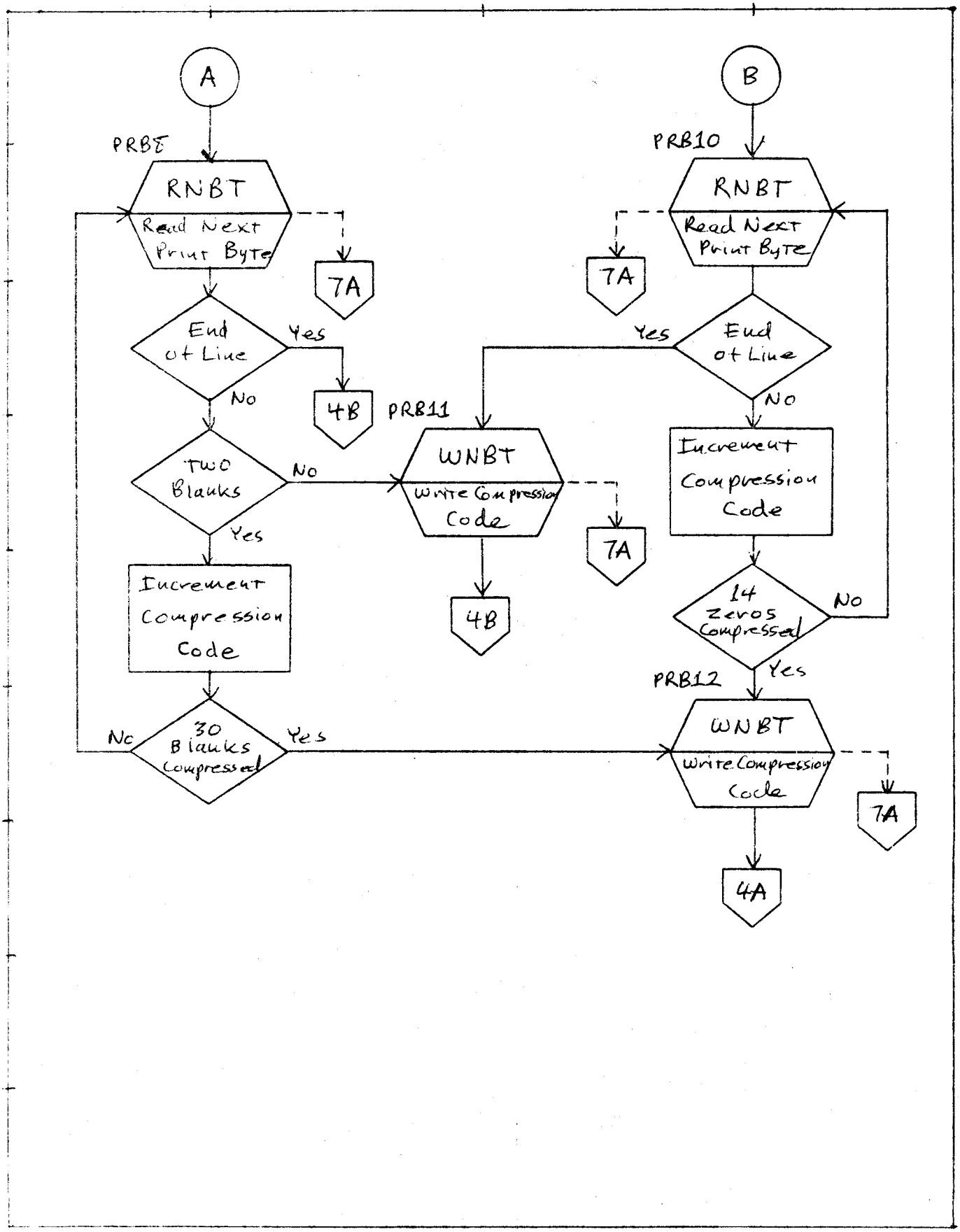
2.5.3 Flow Charts of the Print Processor Routines

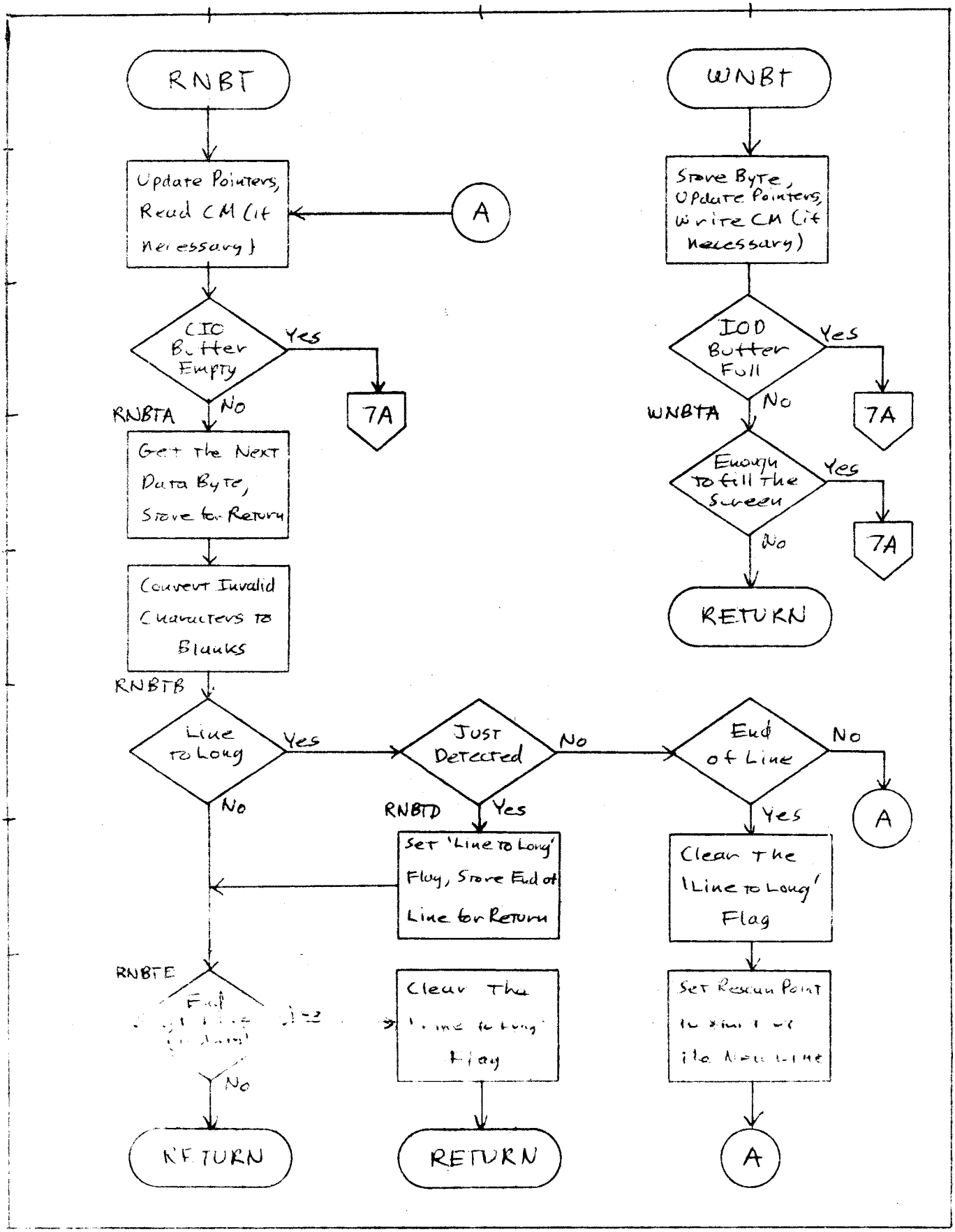

PRTP
A
PRH
Clear the FNT Entry Address
IDLE
Idle Processor, Release Buffer
PR18
Print File Found
Yes
No
Set The FNT Entry Address
PR16
Wait for Previous Buffer to be Released
PR16A
Clear Suspend, End Flags, set heading flag
B
COMP
Process The Command
Yes
Command Input
No
B
PR41
PRNT
Test for Printer Ready
Yes
Command Input
No
PR41B
Printer Ready
Yes
No
2A
MESG
Display 'Printer Not Ready'
COMP
Process The Command
A
COMP
Process The Command
A
No
Test Complete
Yes
PR41A
COMP
Process The Command
A
2A

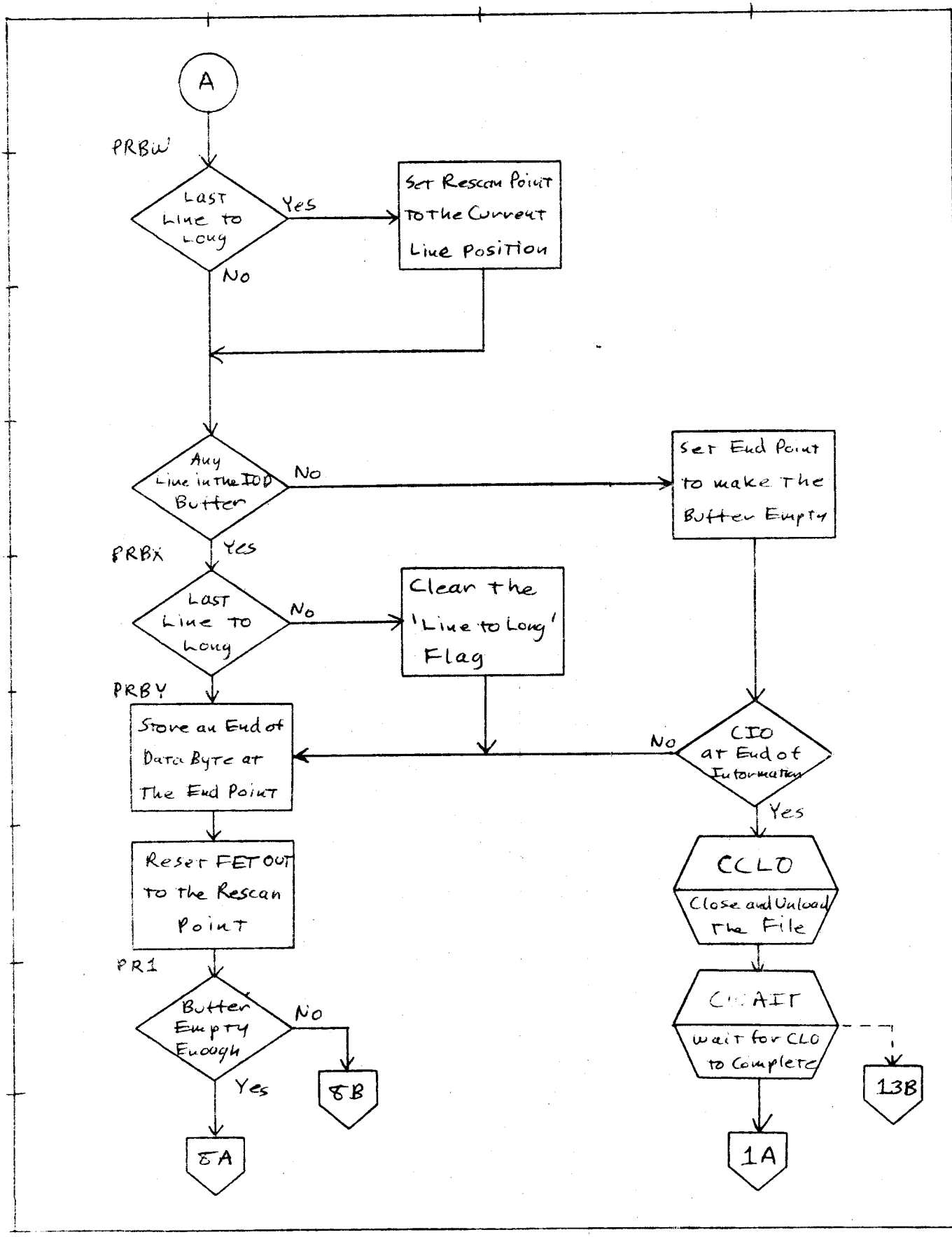

A
PR40
STAL
Assign a new Buffer
Buffer Assigned
Yes
No
Command Input
No
Yes
COMP
Process The Command
1A
PR42
Set up the FET for the Print File
PR43
Command Input
Yes
No
C
B
B
HEADA0
Generate Banner and write to IOD Buffer
D
HEADC2
PRNT
Print The Banner
Command Input
Yes
No
HEADC3
Printer Ready
Yes
No
MESG
Display 'Printer NOT Ready'
COMP
Process The Command
1A, 13B
D
C
COMP
Process The Command
1A, 13B
COMP
Process The Command
1A, 13B
No
Print Complete
Yes
HEADC2A
COMP
Process The Command
1A, 13B
HEADC4
Printed Twice
No
D
Yes
Clear the Heading Flag
3A

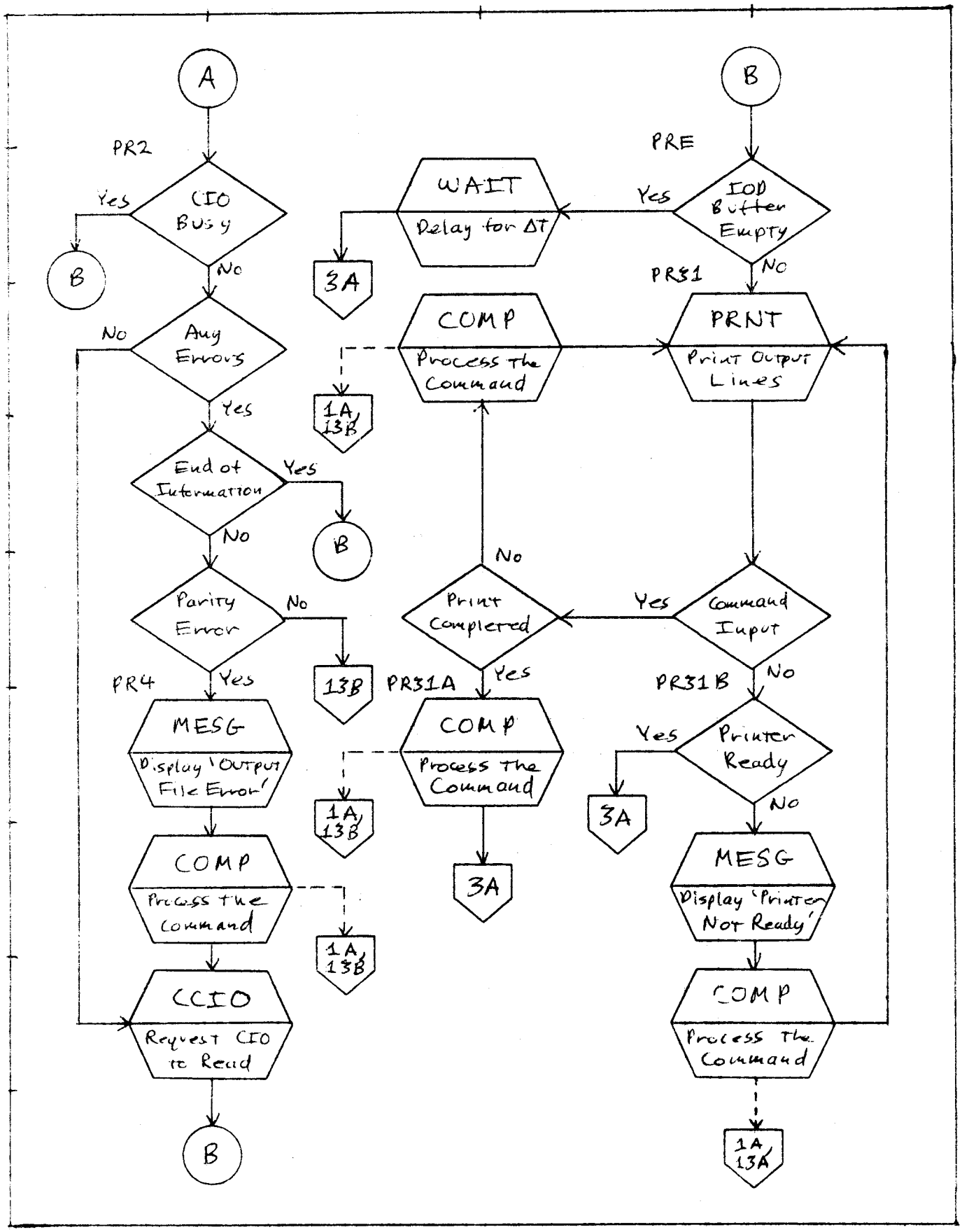

A
PRB
Initialize the Print Data Scan Starting from OUT
B
PRB1
Set the Rescan Point to the Start of a New Line
D
PRB1A
RNBT
Read First Byte of the line
PRB1A
WNBT
Write First Byte of the line
7A
7A
PM Message
YES
NO
4A
Any Lines in IOD Buffer
YES
7A
NO
C
C
PRB2
WNBT
Write PM Message Byte
RNBT
Read Next PM Message Byte
7A
NO
END of Line ?
YES
PRB3
WNBT
Write End of Data Byte
Reset FET OUT to Start of Next Line
MESG
Display PM Message
COMP
Process the Command
1A, 13B
A

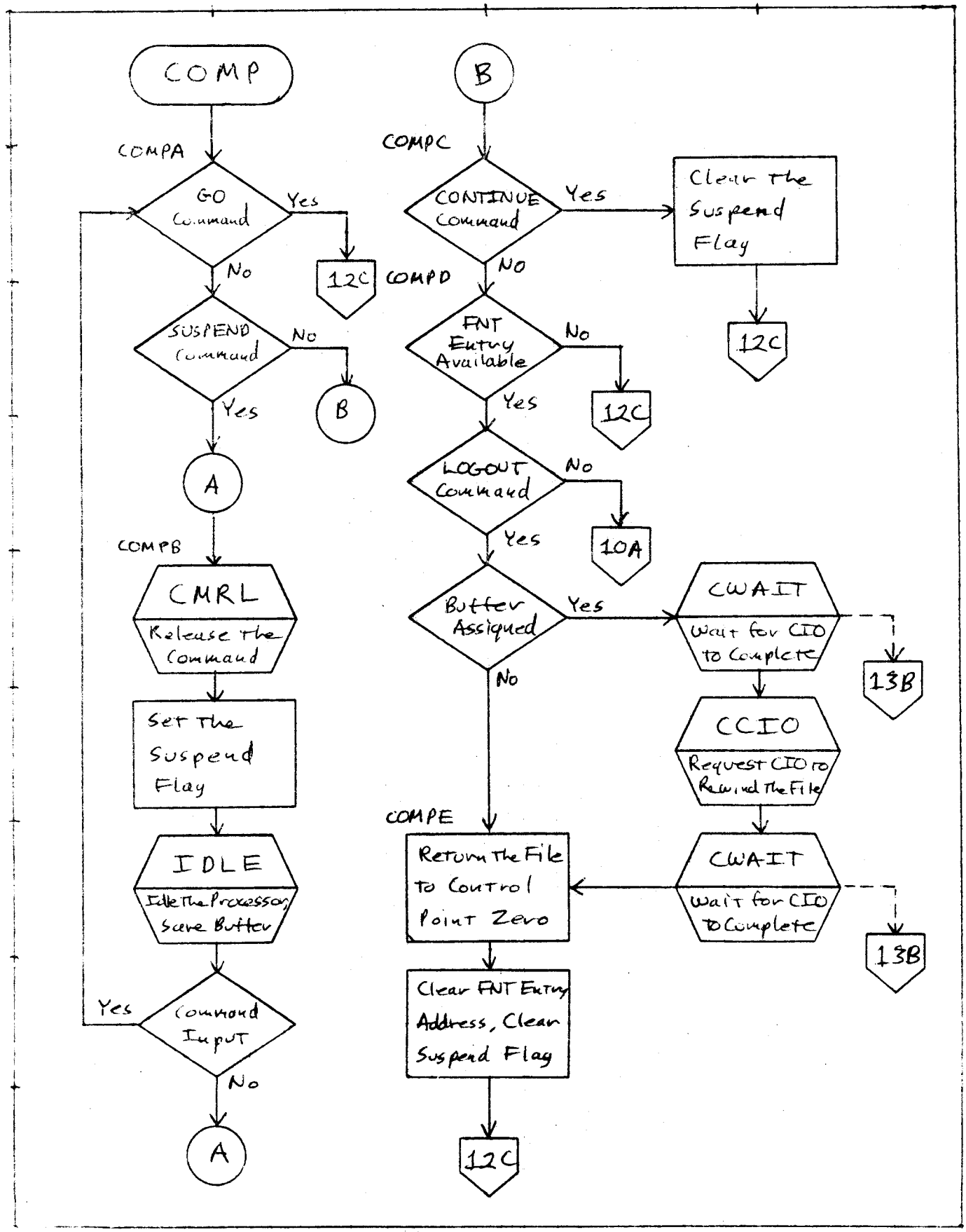

A
PRB4
RNBT
Read Next Print Byte
7A
B
PRB5
END of Line ?
YES
NO
Set Line in IOD Buffer Flag, Set End Point To The end of this line, Set Rescan Pointer to start of new line,
WNBT
Write End of line Byte
7A
PRB7
Set INITIAL Blank Compression Code
YES
PRB6
TWO BLANKS ?
NO
INITIALIZE Read Buffer Pointer, START.
3D
RNBT
Read Next Print Byte
7A
End of Line ?
yes
B
NO
Two Blanks ?
Yes
5A
NO
WNBT
Write Two Blanks
7A
B
TWO Zeros
yes
NO
PRB9
Set INITIAL Zero Compression Code
WNBT
Write Print Byte
7A
A
RNBT
Read Next Print Byte
7A
Two Zeros ?
yes
5B
NO
WNBT
Write Two Zeros
7A
B

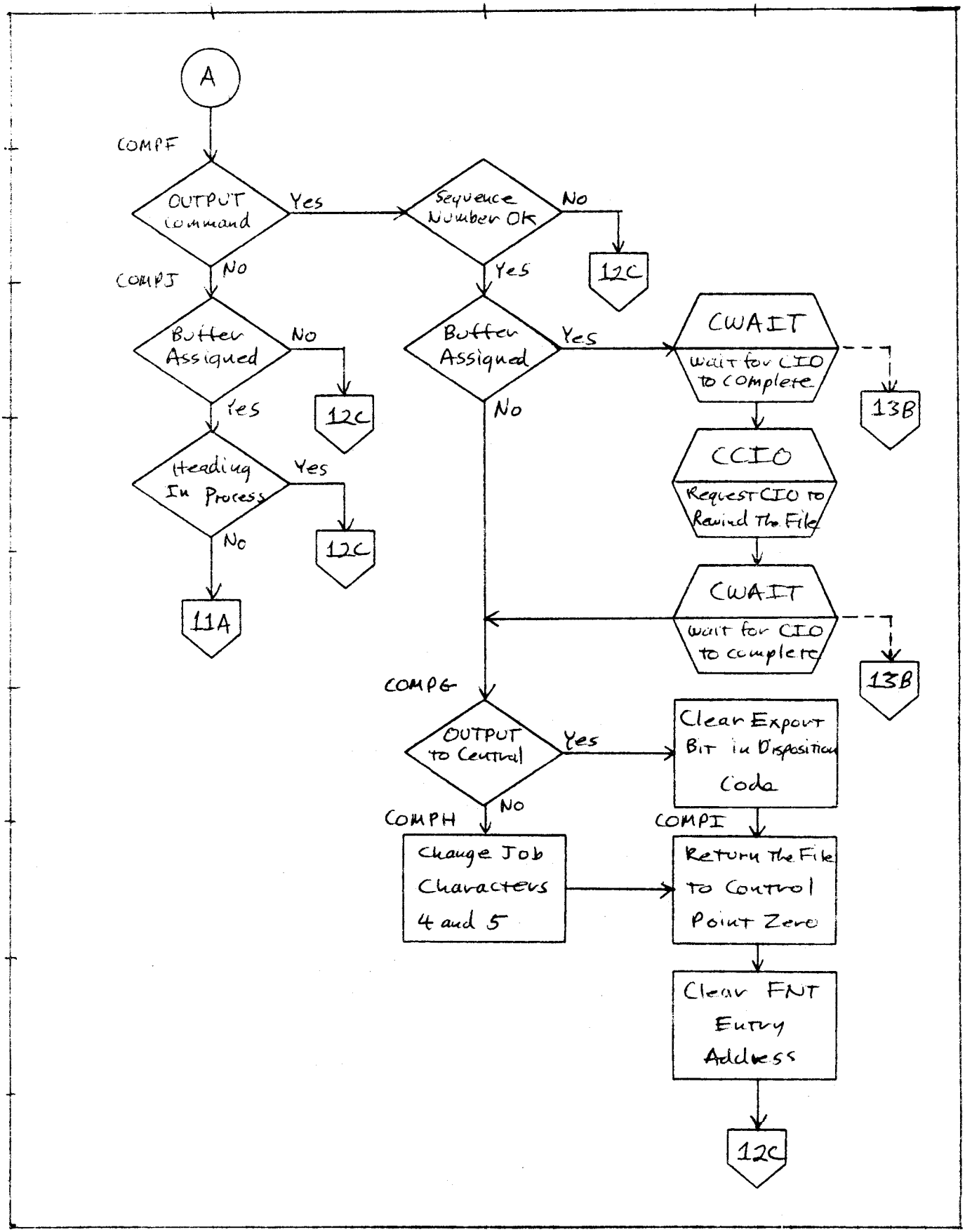

A
PRBE
RNBT
Read Next Print Byte
7A
End of Line
Yes
No
4B
Two Blanks
No
Yes
Increment Compression Code
30 Blanks Compressed
No
Yes
PRB11
WNBT
Write Compression Code
7A
4B
B
PRB10
RNBT
Read Next Print Byte
7A
End of Line
Yes
No
Increment Compression Code
14 zeros Compressed
No
Yes
PRB12
WNBT
Write Compression Code
7A
4A

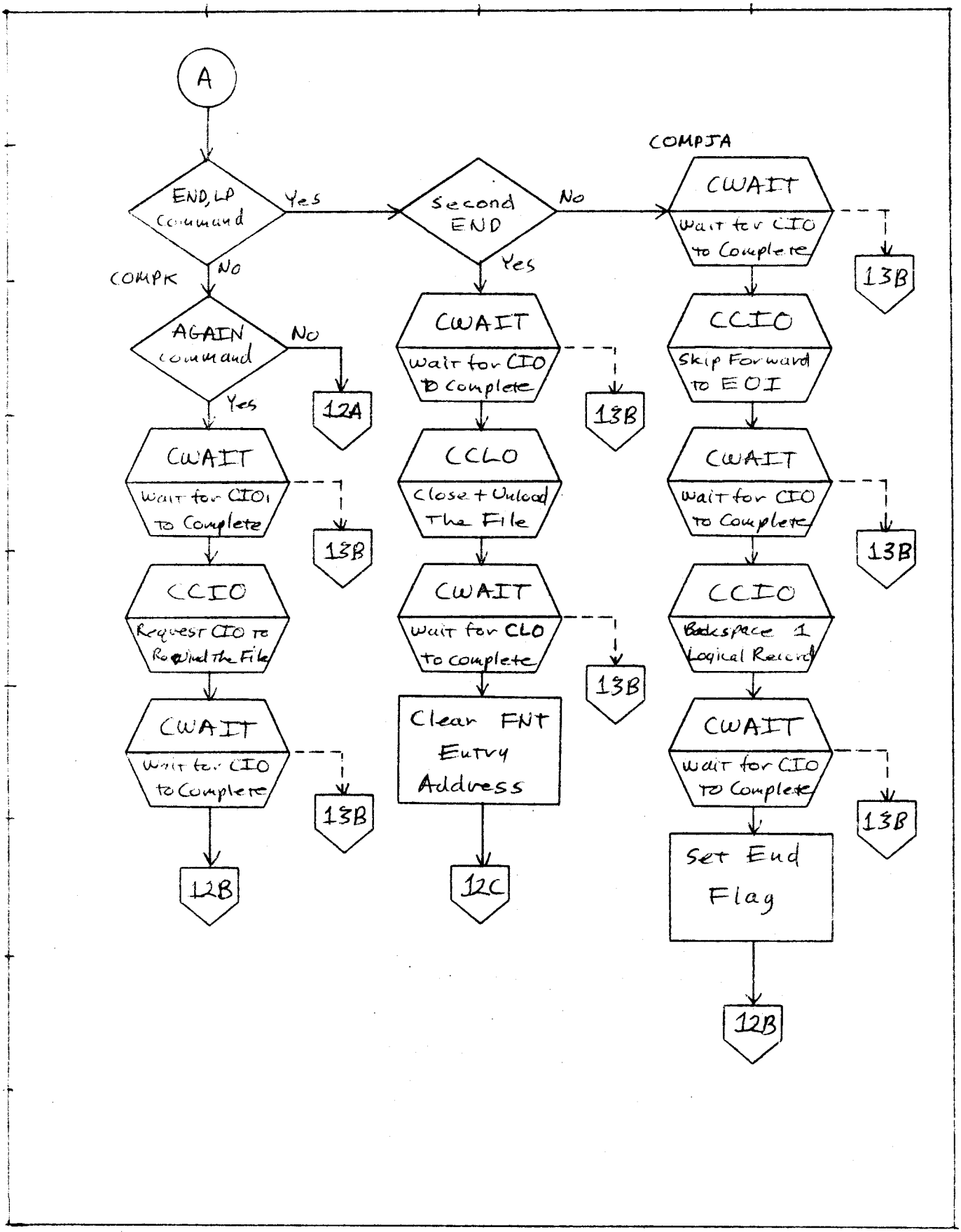

RNBT
Update Pointers, Read CM (if necessary)
A
CIO Butter Empty
Yes
7A
No
RNBTA
Get the Next Data Byte, Store for Return
Convert Invalid Characters to Blanks
RNBTB
Line to Long
Yes
No
Just Detected
No
Yes
RNBTD
Set 'Line to Long' Flag, Store End of Line for Return
End of Line
No
A
Yes
Clear The 'Line to Long' Flag
Set Return Point to start of the New Line
A
RNBTE
End
No
RETURN
Clear The 'Line to Long' Flag
RETURN
WNBT
Store Byte, Update Pointers, Write CM (if necessary)
IOD Butter Full
Yes
7A
No
WNBTA
Enough to fill the Screen
Yes
7A
No
RETURN

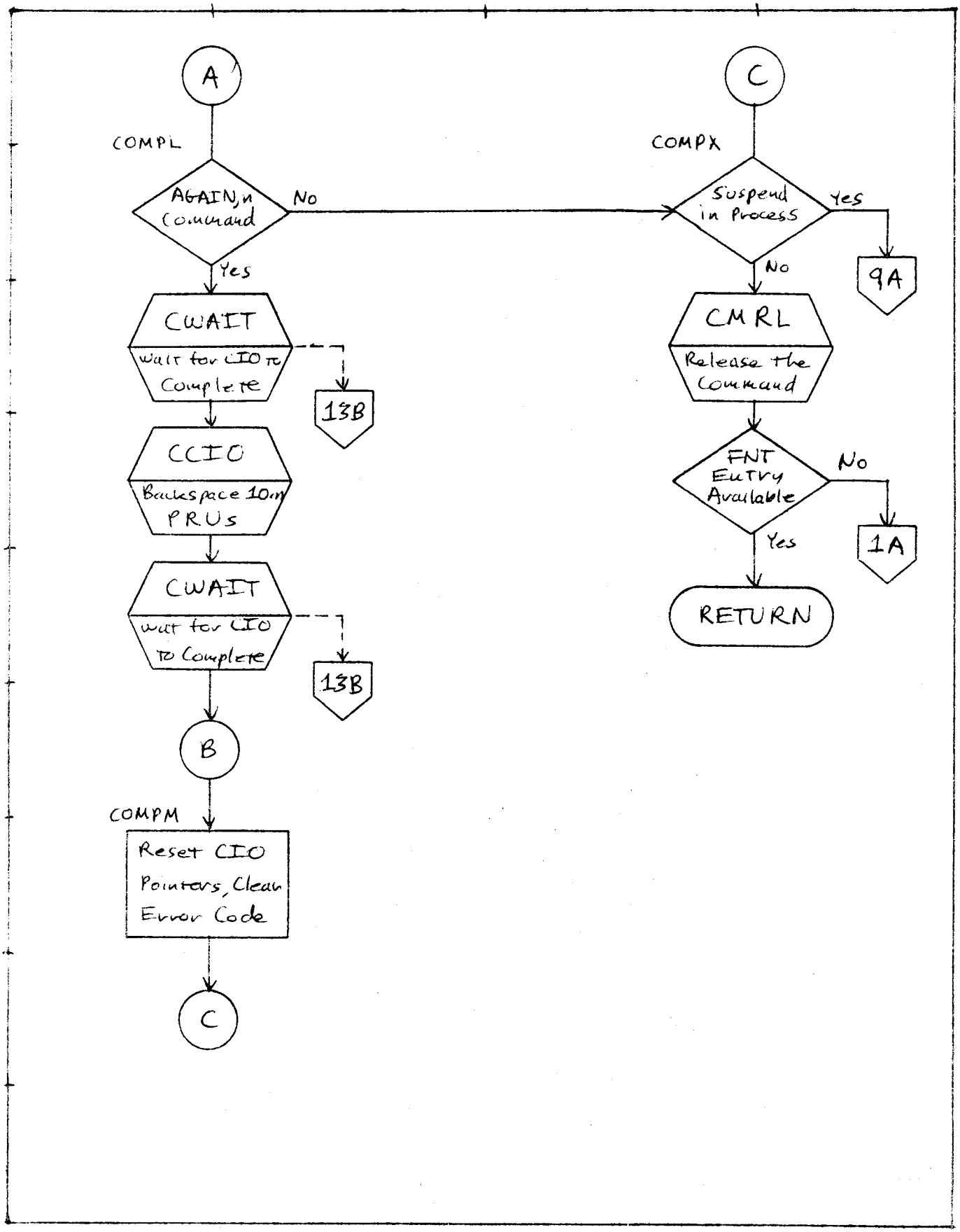

A
PRBW
Last Line to Long
Yes
Set Rescan Point to the Current Line Position
No
Any Line in the IOD Buffer
No
Set End Point to make the Buffer Empty
PRBX
Yes
Last Line to Long
No
Clear the 'Line to Long' Flag
PRBY
Store an End of Data Byte at The End Point
No
CIO at End of Information
Yes
Reset FET OUT to the Rescan Point
CCLO
Close and Unload the File
PR1
Buffer Empty Enough
No
Yes
8B
8A
CWAIT
wait for CLO to Complete
13B
1A

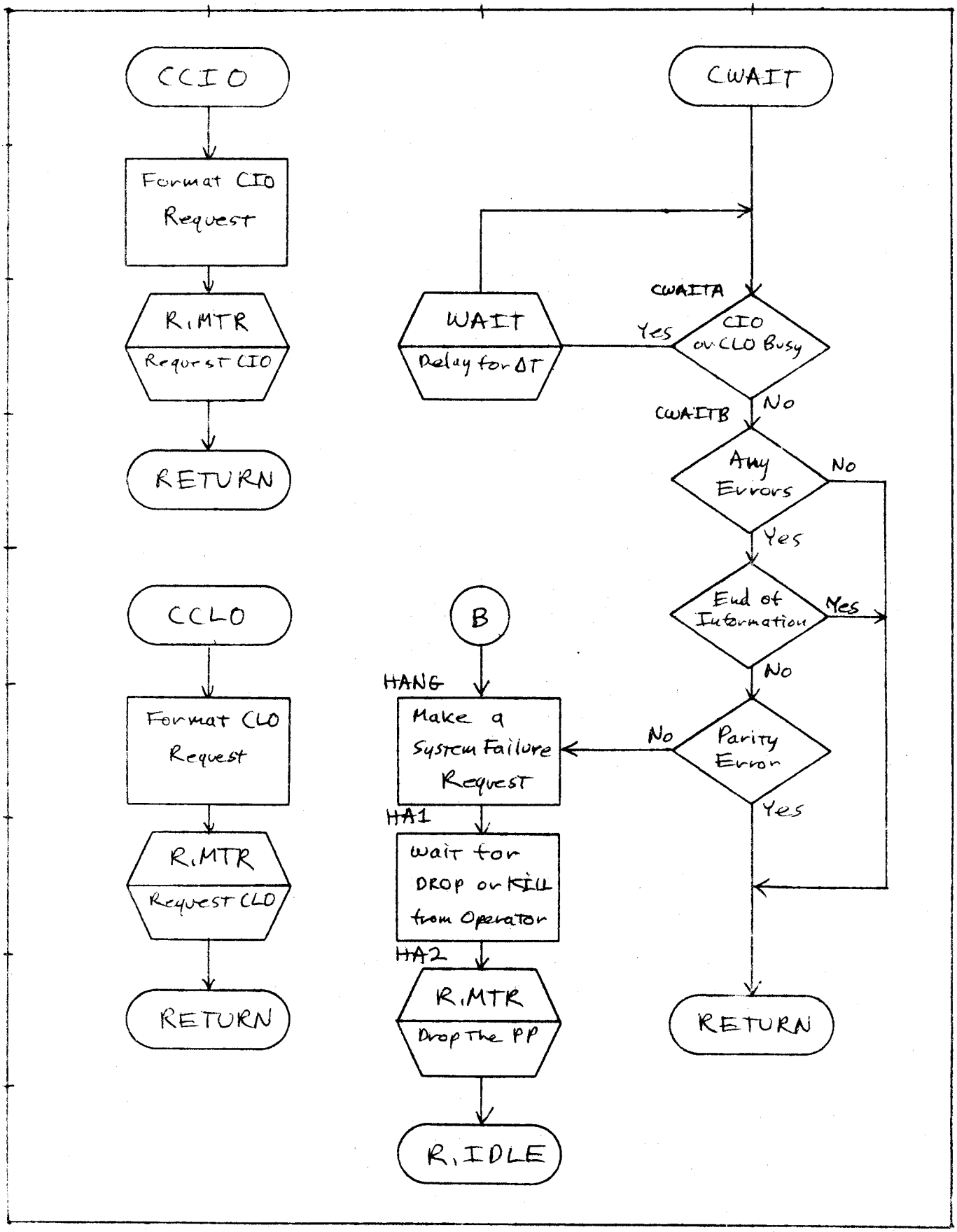

A
PR2
CIO Busy
Yes
B
No
Any Errors
No
Yes
End of Information
Yes
B
No
Parity Error
No
13B
PR4
Yes
MESG
Display 'Output File Error'
COMP
Process the Command
1A, 13B
CCIO
Request CIO to Read
B
WAIT
Delay for ΔT
3A
COMP
Process the Command
1A, 13B
B
PRE
IOD Buffer Empty
Yes
No
PR31
PRNT
Print Output Lines
Command Input
Yes
No
PR31B
Printer Ready
Yes
3A
No
MESG
Display 'Printer Not Ready'
COMP
Process the Command
1A, 13A
Print Completed
No
Yes
PR31A
COMP
Process the Command
3A


COMP
COMPA
GO Command
Yes
No
12C
SUSPEND Command
No
Yes
B
A
COMPB
CMRL
Release the Command
Set The Suspend Flag
IDLE
Idle The Processor, Save Buffer
Yes
Command Input
No
A
B
COMPC
CONTINUE Command
Yes
Clear the Suspend Flag
12C
COMPD
NO
FNT Entry Available
No
Yes
12C
LOGOUT Command
No
Yes
10A
Buffer Assigned
Yes
No
CWAIT
Wait for CIO to Complete
13B
CCIO
Request CIO to Rewind The File
CWAIT
Wait for CIO to Complete
13B
COMPE
Return The File to Control Point Zero
Clear FNT Entry Address, Clear Suspend Flag
12C


A
COMPF
OUTPUT command
Yes
Sequence Number OK
No
12C
No
COMPJ
Yes
Buffer Assigned
No
12C
Yes
Heading In Process
Yes
12C
No
11A
Buffer Assigned
Yes
CWAIT
Wait for CIO to complete
13B
No
CCIO
Request CIO to Rewind The File
CWAIT
Wait for CIO to complete
13B
COMPG
OUTPUT to Central
Yes
Clear Export Bit in Disposition Code
No
COMPH
Change Job Characters 4 and 5
COMPI
Return The File to Control Point Zero
Clear FNT Entry Address
12C


A
END, LP Command
Yes
No
COMPK
AGAIN command
No
Yes
12A
CWAIT
Wait for CIO1 to Complete
13B
CCIO
Request CIO to Rewind the File
CWAIT
Wait for CIO to Complete
13B
12B
Second END
No
Yes
CWAIT
Wait for CIO to Complete
13B
CCLO
Close + Unload The File
CWAIT
Wait for CLO to complete
13B
Clear FNT Entry Address
12C
COMPJA
CWAIT
Wait for CIO to Complete
13B
CCIO
Skip Forward to EOI
CWAIT
Wait for CIO to Complete
13B
CCIO
Backspace 1 Logical Record
CWAIT
Wait for CIO to Complete
13B
Set End Flag
12B


A
COMPL
AGAIN,n Command
No
Yes
CWAIT
Wait for CIO to Complete
13B
CCIO
Backspace 10,n PRUs
CWAIT
Wait for CIO to Complete
13B
B
COMPM
Reset CIO Pointers, Clear Error Code
C
C
COMPX
Suspend in Process
Yes
9A
No
CMRL
Release the Command
FNT Entry Available
No
1A
Yes
RETURN


CCIO
Format CIO Request
R.MTR
Request CIO
RETURN
CWAIT
CWAITA
CIO or CLO Busy
Yes
WAIT
Delay for ΔT
No
CWAITB
Any Errors
No
Yes
End of Information
Yes
No
Parity Error
No
Yes
RETURN
CCLO
Format CLO Request
R.MTR
Request CLO
RETURN
B
HANG
Make a System Failure Request
HA1
wait for DROP or KILL from Operator
HA2
R.MTR
Drop the PP
R.IDLE

2.5.4 Print Processor Save Area and CM Buffer

Save Area: SAV1 - SAV5, direct cells 63B - 67B

SAV1 - Called FNTADD, this cell is used to save the address of the FNT for the currently printing file.

SAV2 - Called SUBRADD, this cell is used to save the return address of the subroutine CWAIT.

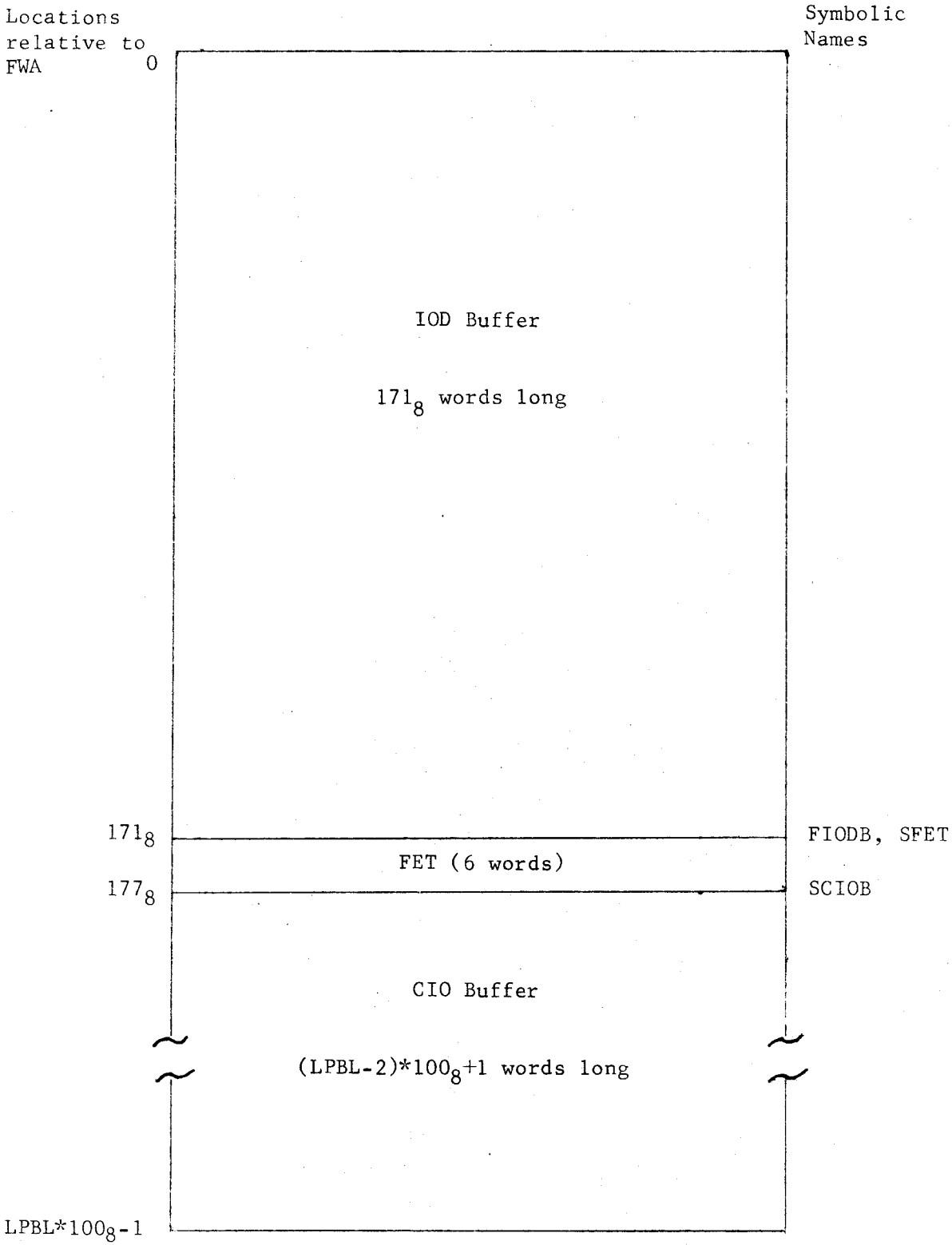
SAV3 - Called SUBRADD1, this cell is used to save the return address of the subroutine COMP.

SAV4 - This cell holds bit flags for COMP.

- 2^0 bit is set while printing is suspended.
- 2^1 bit is set during initialization to prevent executing AGAIN or E,LP instructions.
- 2^2 bit is set after the first executed E,LP so that a second input of E,LP will terminate printing.

SAV5 - This cell is used in various places to hold flags or counts while calling EXEC subroutines.

CM Buffer : First word address = FWA = BAD*100_8+RA
LPBL*100_8 words long ($\geq 300_8$)



2.6 Details of the Message Processor (4MU)

2.6.1 Narrative of the Message Processor Routines

PROGRAM - 4MU
ROUTINE - MSGP

MSGP is the main routine of the message processor overlay and performs the tasks of 1) displaying messages from the central site operator, and 2) analyzing, routing, and, in some cases, executing commands from the remote terminal. When a message is input from the central site operator, FNTS requests EXEC to activate MSGP. When MSGP is activated and no command has been input (or during any period that MSGP is active and waiting for storage or a command), MSGP will periodically look for messages from the central site. When a message is detected, MSGP will call EXEC to interrupt any other display and display the new operator message, and then wait for a command. All commands that are entered are initially locked onto MSGP so that Syntax analysis and verification can be performed. (See Section 2.6.2 for a detailed description).

If the command is not recognized, MSGP calls EXEC to display the message "FORMAT ERROR" and wait until a new command is entered. If the command is recognized and is executed by another processor, MSGP will reset the command lock to the processor which will execute the command, and then go call IDLE. Special cases are GO and LOGOUT. For GO, MSGP will set the command lock to the value of the display lock thereby routing the command to the processor currently locked into the display. (If no processor is locked onto the display, then the command lock is automatically cleared).

The first occurrance of LOGOUT causes MSGP to request a line logout and route the command as LOGOUT to processor 1. All subsequent LOGOUT commands (created by IODC for every attempted I/O call made by EXEC after the logout request is made) are alternately routed to processor 1 and 2 until EXEC finally requests a line drop. Those commands that are recognized as being executed by MSGP are given below.

H - MSGP calls the internal subroutine HGEN to generate the display and then waits until a new command is entered.

B - MSGP calls the internal subroutine BGEN to generate the display and then waits until a new command is entered.

MESSAGE - MSGP reformats and displays the message (to start with "M", only), requests assignment of the central display, and then waits until the display is assigned and clear. When this is true, the message is written to the central site and the display request is cleared.

DROP - MSGP will scan the FNT and, if necessary, the control point area and drop the requested job. Jobs still in the input queue have their RBT chains released, while jobs at control points have their operator drop flags set.

OUTPUT - MSGP will scan the FNT and, if the file is not already printing, will divert the requested file by either changing the disposition to local, or the 4th and 5th job characters to those specified. If the file is currently printing, MSGP will lock the command onto the print processor.

If either DROP or OUTPUT reference a file which is not available to the terminal, MSGP will call EXEC to display the message "FILE NOT FOUND" and then wait until a new command is input. After any successful execution of MESSAGE, DROP or OUTPUT (for a file that was not printing), MSGP will either call IDLE or automatically generate an H or B display. The latter path is taken if MSGP was previously displaying an H or B display when the command was entered.

PROGRAM - 4MU

SUBROUTINE - COPM

COPM checks for a new message from the central site operator and returns if none is found. Otherwise, COPM formats the message and then transfers into MSGP to display the message.

PROGRAM - 4MU
SUBROUTINE - HGEN

HGEN calls an internal version of STAL to obtain a CM buffer and then scans the FNT. The highest priority input and output file names (up to 44) associated with the requesting terminal are stored within the PP memory, and then sorted (in order of priority) into the CM buffer to create the H display. EXEC is called to display the buffer and wait until a new command is entered.

PROGRAM - 4MU
SUBROUTINE - BGEN

BGEN calls an internal version of STAL to obtain a CM buffer and then scans the control point areas. Data from each area (except that at which E/I 200 is running) is stored in the CM buffer to create the B display. EXEC is called to display the buffer and wait until a new command is entered.

PROGRAM - 4MU
SUBROUTINE - STAL

STAL is an internal version of the STAL in EXEC. STAL calls EXEC to display the waiting-for-storage message without processor cycling. Called by HGEN and BGEN.

PROGRAM - 4MU
SUBROUTINE - DCHS

DCHS calls R.DCH to drop the FNT/FST channels assigned by MSGP before any FNT scan. Called by MSGP.

PROGRAM - 4MU
SUBROUTINE - SDIS

SDIS checks a PP word and guarantees both display coded characters are between 01B and 57B. Called by HGEN.

PROGRAM - 4MU
SUBROUTINE - GETC

GETC gets a single character from an input command and advances the character pointer. Called by MSGP during command syntax analysis.

PROGRAM - 4MU
SUBROUTINE - BIDC

BIDC converts a 9-bit binary number into 3 display coded decimal digits with high order zero blanking. Called by HGEN.

PROGRAM - 4MU
SUBROUTINE - SPRI

SPRI scans the list of file names set up by HGEN, selects the next highest priority file name, and writes the name into the CM buffer. Called by HGEN.

PROGRAM - 4MU
SUBROUTINE - BIOC

BIOC converts an 18-bit binary number to 6 display coded octal digits replacing high order zeroes with periods. Called by BGEN.

PROGRAM - 4MU
SUBROUTINE - SE0B

SE0B stores an end-of-buffer code over the last carriage return of an H or B display. Called by HGEN and BGEN.

PROGRAM - 4MU
SUBROUTINE - WWRD

WWRD writes a given number of CM words to a CM buffer. Called by HGEN and BGEN.

PROGRAM - 4MU
SUBROUTINES - LCIE, LCLM

LCIE and LCLM are subroutines to load words from the line status table entry.

PROGRAM - 4MU
SUBROUTINES - SCIE, SCLM

Subroutines to store words into the line status table entry.

2.6.2 Syntax Analysis Section of Message Processor

Commands entered into EXPORT/IMPORT 200 from the terminal keyboard are analyzed and validated by the message processor overlay, 4MU. Analysis is performed by interpretively executing a syntax analysis program that appears as a table within the overlay. Each "instruction" in the syntax table occupies one PP word. The high order octal digit of each instruction (referred to as X) specifies the instruction type, while the second octal digit (referred to as Y) either sub-defines the instruction type or is used as an address or data. The two low order octal digits of each instruction (referred to as ZZ) are used either as an address or as data.

In addition to the instructions, there are 7 working registers and an input and output stream associated with the interpretive "computer." The input stream consists of the characters input from the terminal, while the output stream consists of data, extracted by the analysis program, that will be used by the processor to which the command was directed. The 7 working registers and their uses are as follows:

REGISTER NAME	USE
SYNA	Contains the address, relative to the beginning of the syntax table, of the "instruction" currently being executed.
CHRN	Contains the address, relative to the first character of the input stream, of the next character in the input stream to be investigated.
MSGA	Contains the address, relative to the first PP word of the output stream, of the next location to receive some output data.

SYNC	Temporarily saves the ZZ portion of certain instructions.
SYNS	Contains the branch address, relative to the beginning of the syntax table, that was specified by the last break point.
CHRS, MSGS	Act as a one-level push-down store for CHRN, and MSGS (respectively) when a break point instruction is encountered in the syntax table.

The syntax table instructions are divided into three catagories; namely, 1) breakpoints, 2) character operators, and 3) program stops.

A breakpoint instruction establishes a delayed conditional program branch which is executed if any character operators, following in the program flow, detect an error condition. When the breakpoint is encountered, the branch address is saved in SYNS and the current contents of CHRN and MSGA are saved in CHRS and MSGS, respectively. Execution of the branch will cause CHRN and MSGA to be reset to their values when the breakpoint was encountered (as saved in CHRS and MSGS) and SYNA to be changed to the specified branch address (as saved in SYNS). Therefore, a given breakpoint remains in effect until the next breakpoint is encountered in the program flow. If the branch address points to the first location of the table (SYNS=0) when the branch is executed, an error stop occurs, the message:

FORMAT ERROR

is issued, and the interpreter waits until a new keyboard entry is made. When the new entry is received, the interpreter restarts the syntax program from the beginning.

The character operator instructions examine the input stream and deposit data in the output stream. If a requested operation detects an erroneous character in the input stream, the program flow branches to the address given in the last encountered breakpoint instruction.

A program stop instruction terminates the syntax program and indicates that the command has been recognized. The data portion of this instruction provides the command number and number of the processor to receive the command.

If the entire command syntax is considered to be a binary tree structure, then the breakpoints would appear at each node, the character operators along each branch, and the program stops at the end of a branch which recognizes a command. For those nodes which indicate a syntax error along their alternate path, the breakpoint branch address points to the beginning of the tree.

Each of the allowed syntax table "instructions" has been defined as a macro. If a tag is included when the macro is used, this will become the COMPASS tag of the PP word defined by the macro. These tags are referenced by the breakpoint macro to define the required branch address. The first location of the table has been pre-defined as a breakpoint instruction with a tag of SYNT. This instruction is never executed and the "program" always begins with the second (SYNA=1) instruction. The syntax table "instruction" definitions are given in Table I.

MACRO	INSTRUCTION	OPERATION
BRK LOC	X=1 YZZ=LOC-SYNT	Breakpoint. When encountered: CHRN → CHRS MSGA → MSGS YZZ → SYNS SYNA+1 → SYNA When executed, if SYNS=0, the program stops and a format error is declared; otherwise: CHRS → CHRN MSGS → MSGA SYNS → SYNA
CMP A	X=0 Y=0 ZZ=display coded value of the character A	Character operator. If the next input stream character equals ZZ, then: CHRN+1 → CHRN SYNA+1 → SYNA otherwise, execute the breakpoint branch.
DLM	X=0 Y=1 ZZ not used	Character operator. If the next input stream character is a delimiter (blank or comma), then: CHRN+1 → CHRN SYNA+1 → SYNA otherwise, execute the breakpoint branch.
ALN N	X=0 Y=2 ZZ=N	Character operator. If the next N input stream characters are all alphanumeric, then: CHRN+N → CHRN SYNA+1 → SYNA otherwise, execute the breakpoint branch.
NUM N	X=0 Y=3 ZZ=N	Character operator. If the next N input stream characters are all octal digits, then form the binary value of the N octal digits and store the result in the next output stream word, and: CHRN+N → CHRN MSGA+1 → MSGA SYNA+1 → SYNA otherwise, execute the breakpoint branch.
SAV N	X=0 Y=4 ZZ=N	Character operator. Store the next N input stream characters, right adjusted, in the next N output stream bytes. CHRN+N → CHRN MSGA+N → MSGA SYNA+1 → SYNA

MACRO	INSTRUCTION	OPERATION
BKS N	X=0 Y=5 ZZ=N	Character operator. Backspace the input stream pointer. CHRN-N ⟶ CHRN SYNA+1 ⟶ SYNA
TRM	X=0 Y=6 ZZ not used	Character operator. If the next input stream character is a terminator (00 code or period), then: CHRN+1 ⟶ CHRN SYNA+1 ⟶ SYNA otherwise, execute the breakpoint branch.
FIN N,M	X=2 Y=N ZZ=M	Program stop. Y is the processor number and ZZ is the command number.

When the program is started, the registers are initialized as follows:

SYNA = SYNS = 1
MSGA = MSGS = 1
CHRN = CHRS = 0

The input stream is a 15 PP word table starting at MSGB. The output stream is a 6 PP word table starting at MSGT. When a program stop instruction is executed, the command number is stored in MSGT, while any output data is stored by the interpreter starting at MSGT+1. A list of the recognized commands and their allowed syntax is given in Table II.

H { Δ . }

B { Δ . }

{ R / READ } { Δ . }

{ G / GO } { Δ . }

{ S / SUSPEND } { Δ . }

{ C / CONTINUE } { Δ . }

{ E / END } { b , } { CR { Δ . } LP { Δ . } }

{ A / AGAIN } { Δ . b , } { n_0 / n_1n_0 / $n_2n_1n_0$ } { Δ . }

{ D / DROP } { b , } { $c_1c_2c_3c_4c_5c_6c_7$ / c_6c_7 } { Δ . }

{ O / OUTPUT } { b , } { $c_1c_2c_3c_4c_5c_6c_7$ / c_6c_7 } { Δ . b , } y_1y_2 { Δ . }

{ M / MESSAGE } { b , } $c_1c_2c_3,---,c_n$

LOGOUT

n_i = octal digit
c_i = Alphanumeric character
b = blank
Δ = send indicator (00 code)
{ = a choice between the bracketed characters

The commands H, B, DROP, and MESSAGE are always processed by 4MU. READ and END CR are sent to the reader processor overlay, 4RU, while SUSPEND, CONTINUE, END LP, and AGAIN are sent to the printer processor overlay, 4PU. GO is sent to the processor currently calling the EXEC subroutine MESG. LOGOUT causes a line logout request and is then sent alternately to 4RU and 4PU. (When a line logout has been requested, IODC will interpret all calls for I/O and immediately return a read of the command LOGOUT so as to facilitate the line drop request.) OUTPUT is processed by 4MU if the file has not already been sent to 4PU; otherwise, 4PU processes the command. For both DROP and OUTPUT, 4MU does the required scan of the FNT and, if necessary for DROP, the control point area to see if the specified file is actually in the system. Since only the sequence number need be specified for these commands, if the full job name is given, the first five characters are ignored. Command numbers and any required data are stored in the line message buffer and are automatically loaded into the PP by EXEC (starting at location COMD) before a return is made to the appropriate processor. The data is formatted exactly as deposited in the output stream (MSGT) by the syntax analysis program.

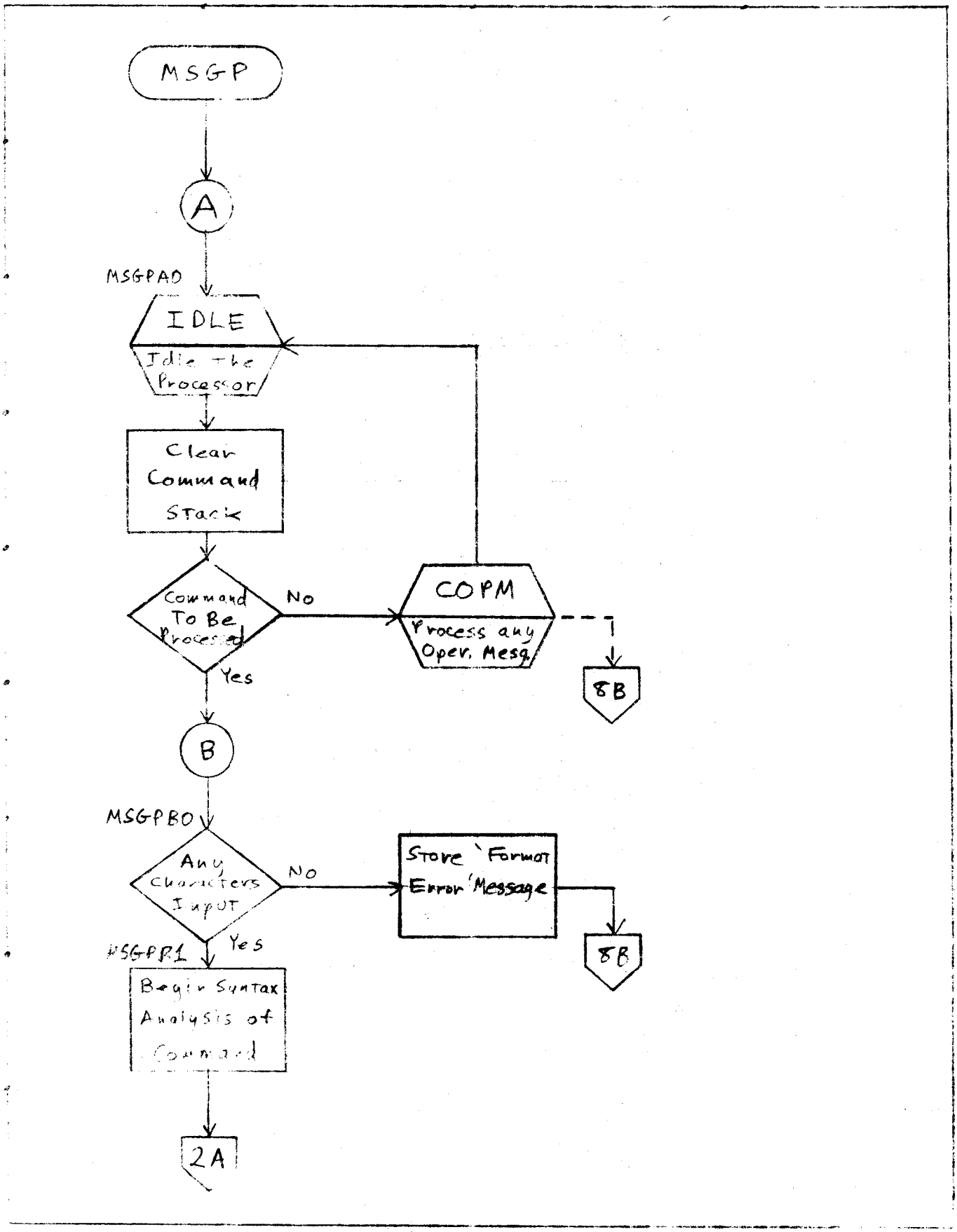
Table III gives the command number and data transmitted to the processor for each command.

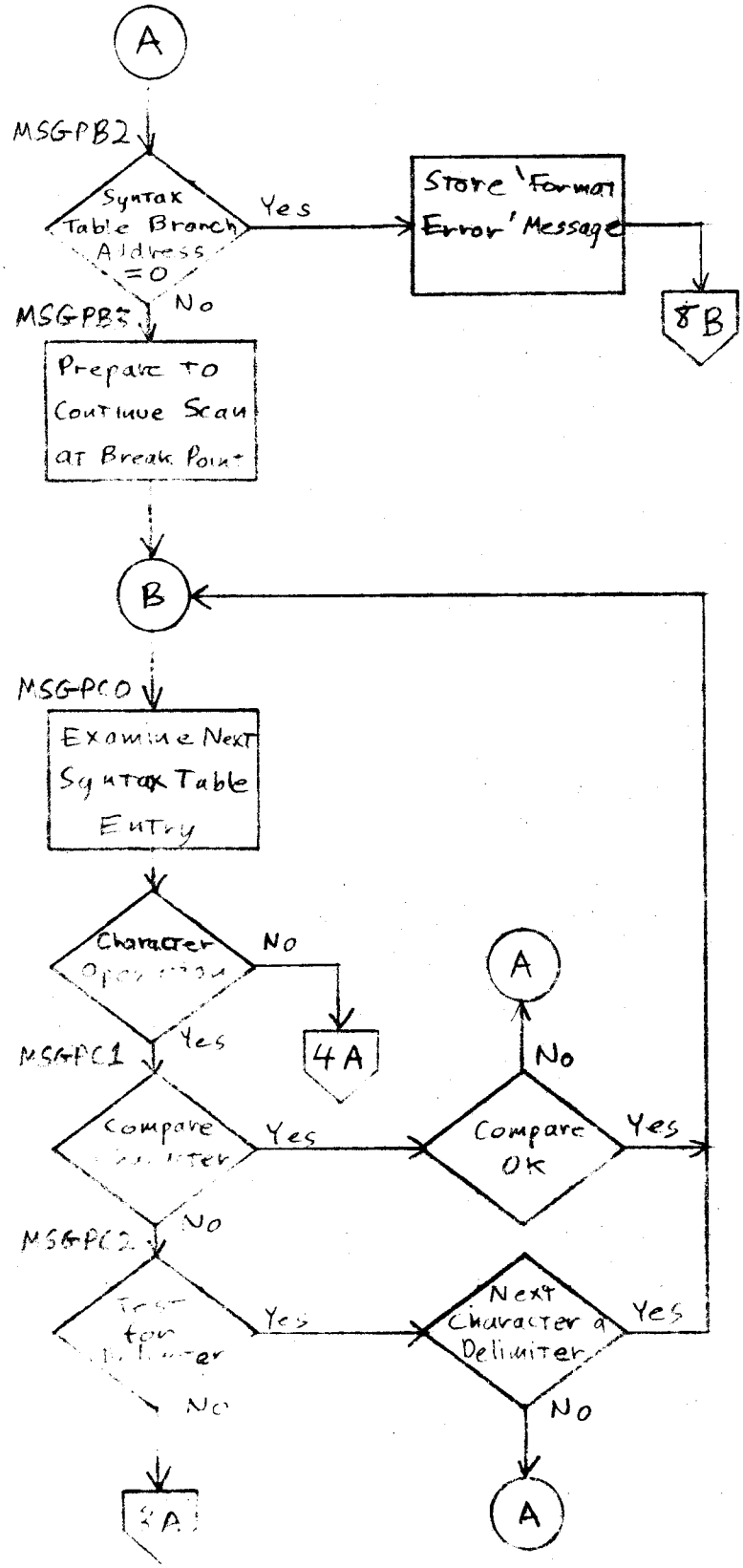
TABLE III

Command	Processor	Command No.$_8$ {LOC. COMD}^{8}	Data {LOC. COMD+1, +2, ---}
H	Message	1	none
B	Message	2	none
READ	Reader	3	none
GO	Reader, Printer	0	none
SUSPEND	Printer	4	none
CONTINUE	Printer	5	none
END, CR	Reader	6	none
END, LP	Printer	7	none
AGAIN	Printer	10	none
AGAIN, n	Printer	11	n
DROP, c_6c_7	Message	12	c_6, c_7
OUTPUT, c_6c_7	Message, Printer	13	c_6, c_7
OUTPUT, c_6c_7, y_1y_2	Message, Printer	14	c_6, c_7, y_1, y_2
QUIT	Message	30	none
MESSAGE, c_1c_2---c_n	Message	55	c_1, c_2, ---c_n, 1
LOGOUT	Reader, Printer	70	none

2.6.3 Flow Chart of the Message Processor Routines

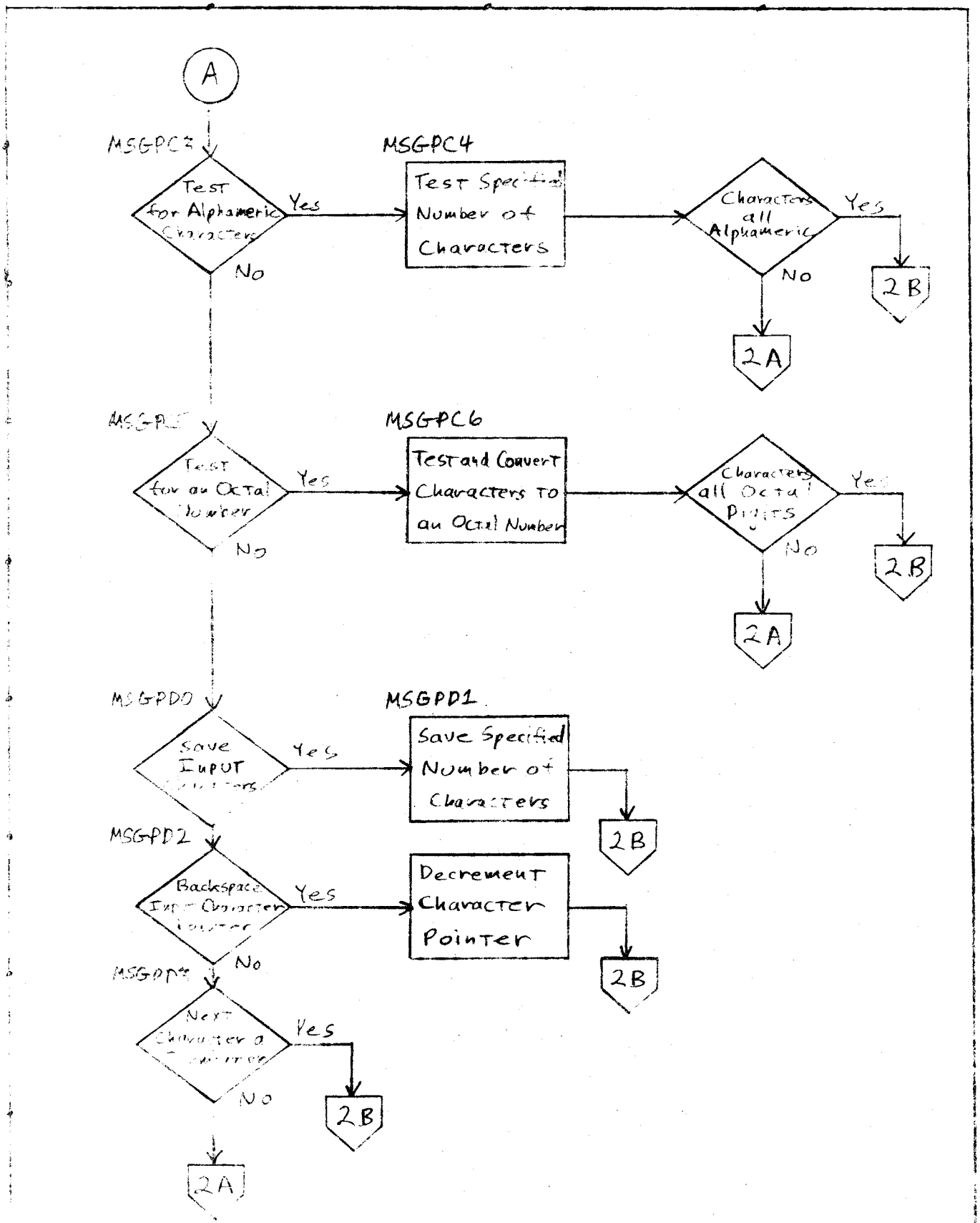
4 MU

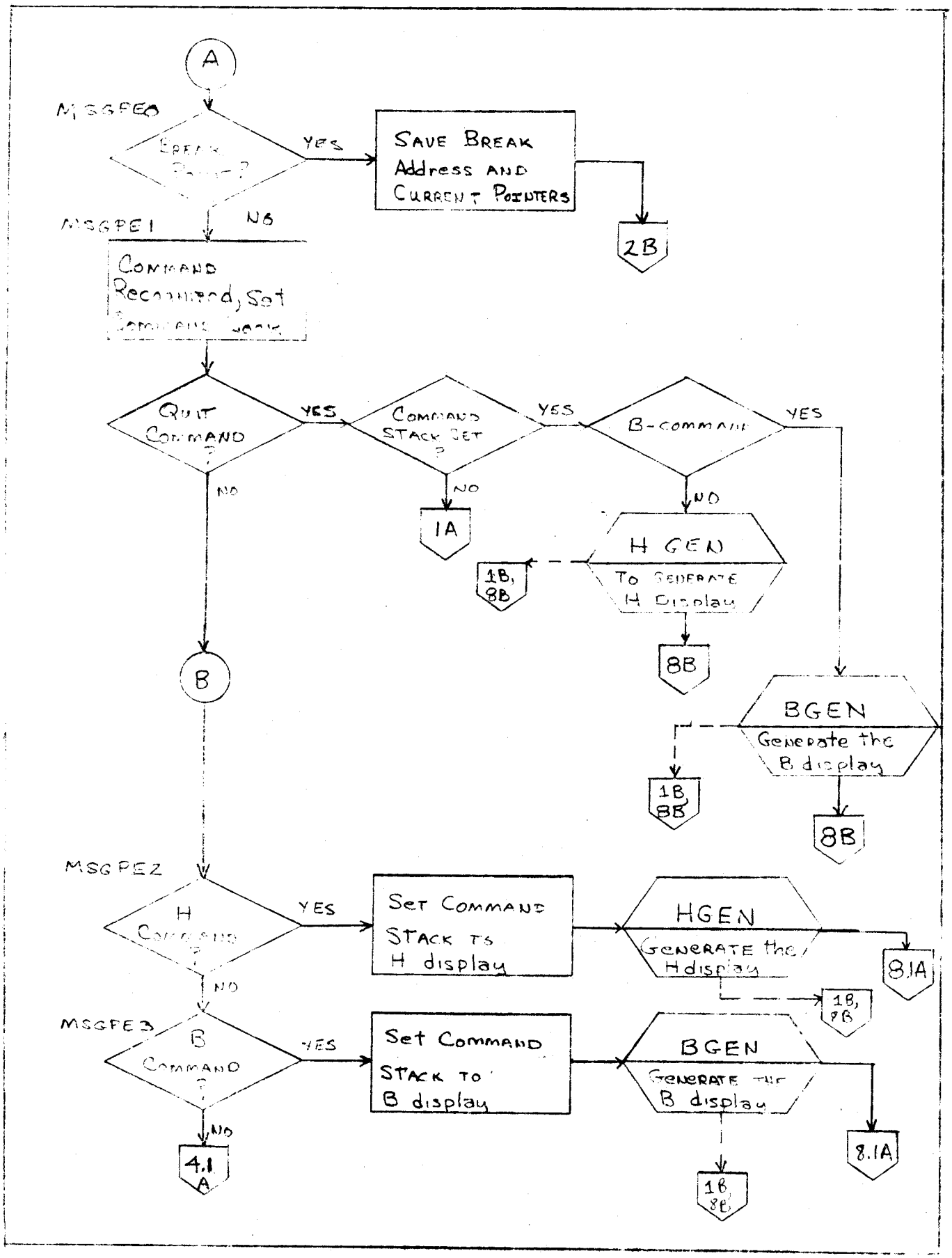

MSGP
A
MSGPAD
IDLE
Idle the Processor
Clear Command Stack
Command To Be Processed
No
COPM
Process any Oper. Mesg
8B
Yes
B
MSGPB0
Any characters Input
No
Store 'Format Error' Message
8B
MSGPR1
Yes
Begin Syntax Analysis of Command
2A

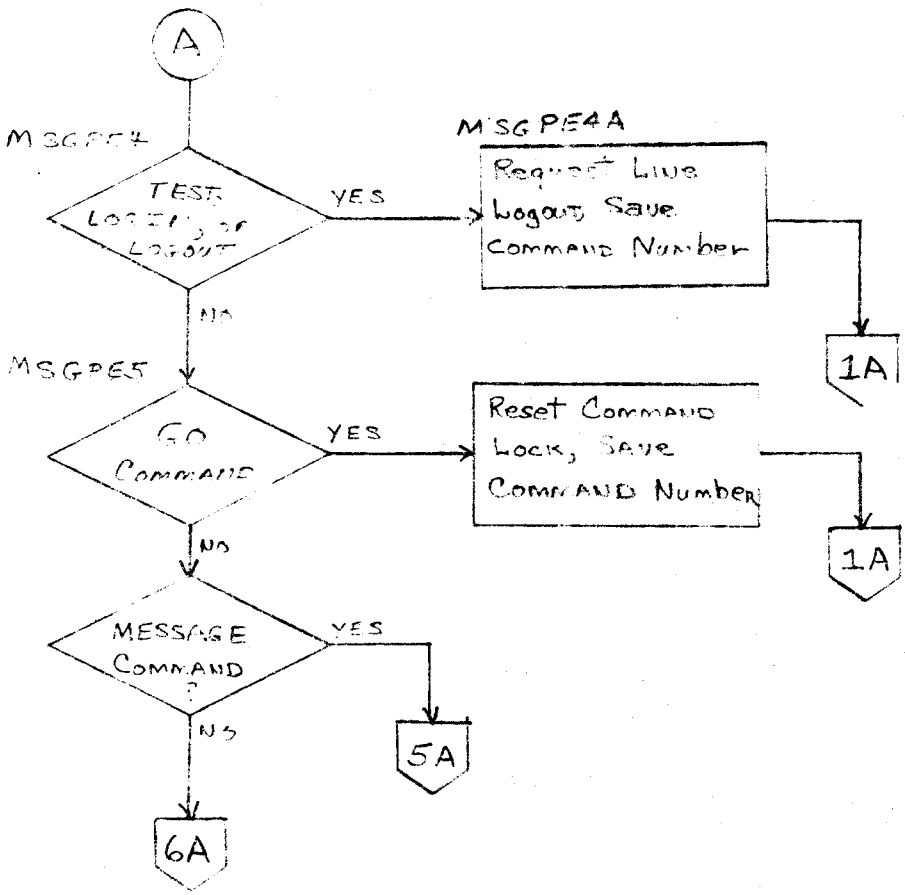

A
MSGPB2
Syntax Table Branch Address =0
Yes
Store 'Format Error' Message
8B
No
MSGPB3
Prepare to Continue Scan at Break Point
B
MSGPC0
Examine Next Syntax Table Entry
Character
No
4A
Yes
MSGPC1
Compare
Yes
A
No
Compare OK
Yes
MSGPC2
Test for Delimiter
Yes
Next Character a Delimiter
Yes
No
No
2A
A

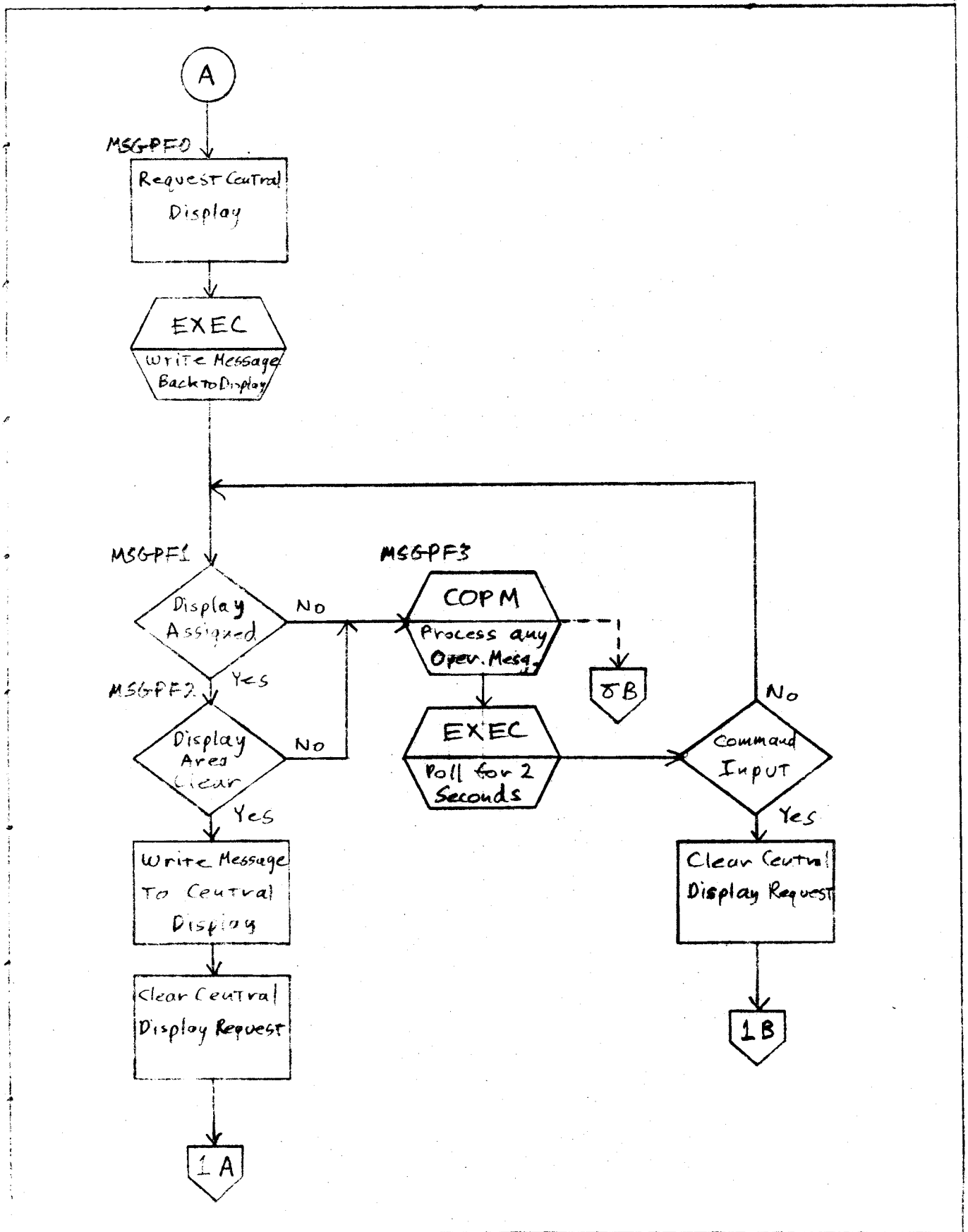
4110

Page 3

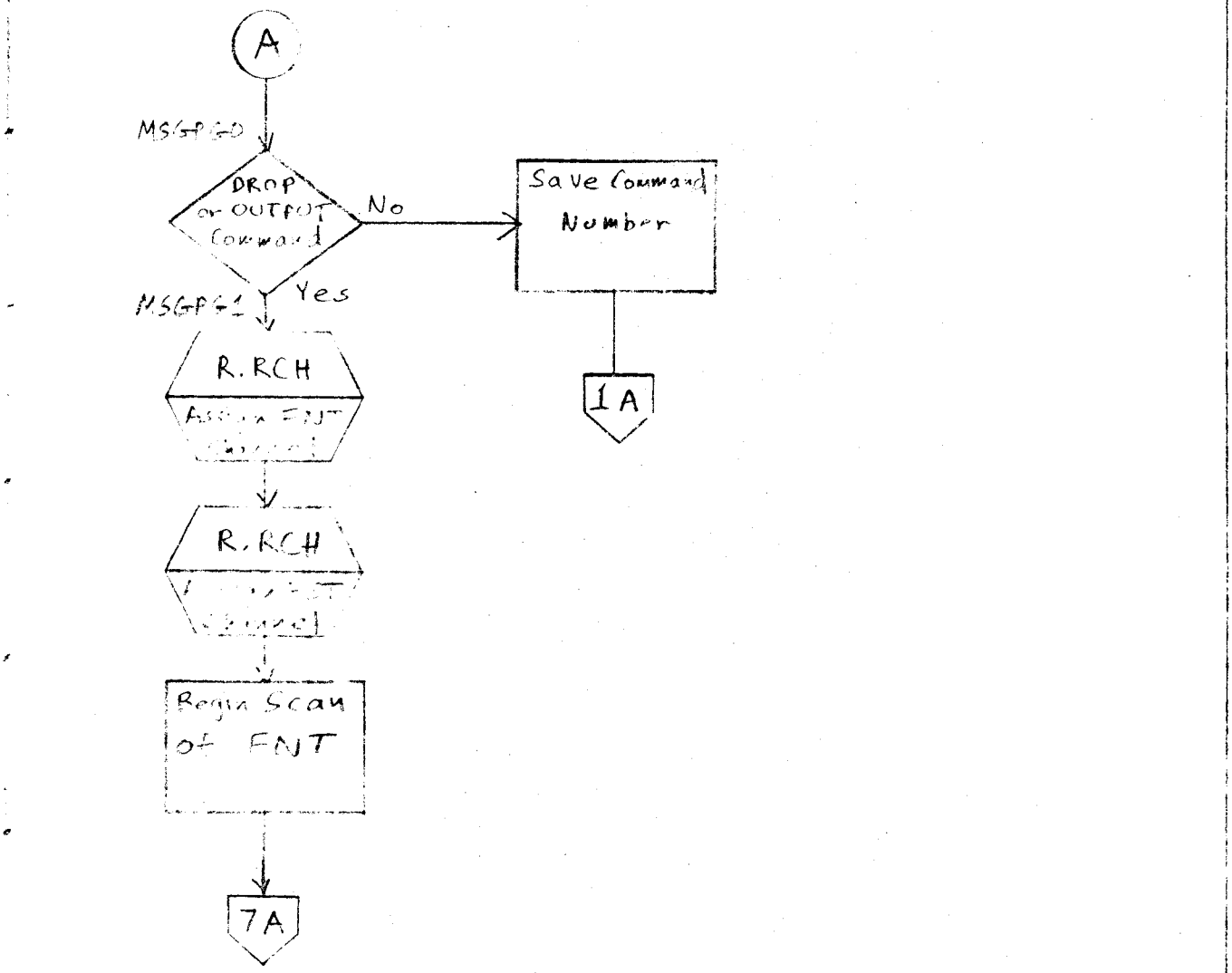

A
MSGPC3
Test for Alphameric Characters
Yes
No
MSGPC4
Test Specified Number of Characters
Characters all Alphameric
Yes
No
2B
2A
MSGPC5
Test for an Octal Number
Yes
No
MSGPC6
Test and Convert Characters to an Octal Number
Characters all Octal Digits
Yes
No
2B
2A
MSGPD0
Save Input Characters
Yes
MSGPD1
Save Specified Number of Characters
2B
MSGPD2
Backspace Input Character Pointer
Yes
Decrement Character Pointer
2B
No
MSGPD3
Next Character a Terminator
Yes
2B
No
2A

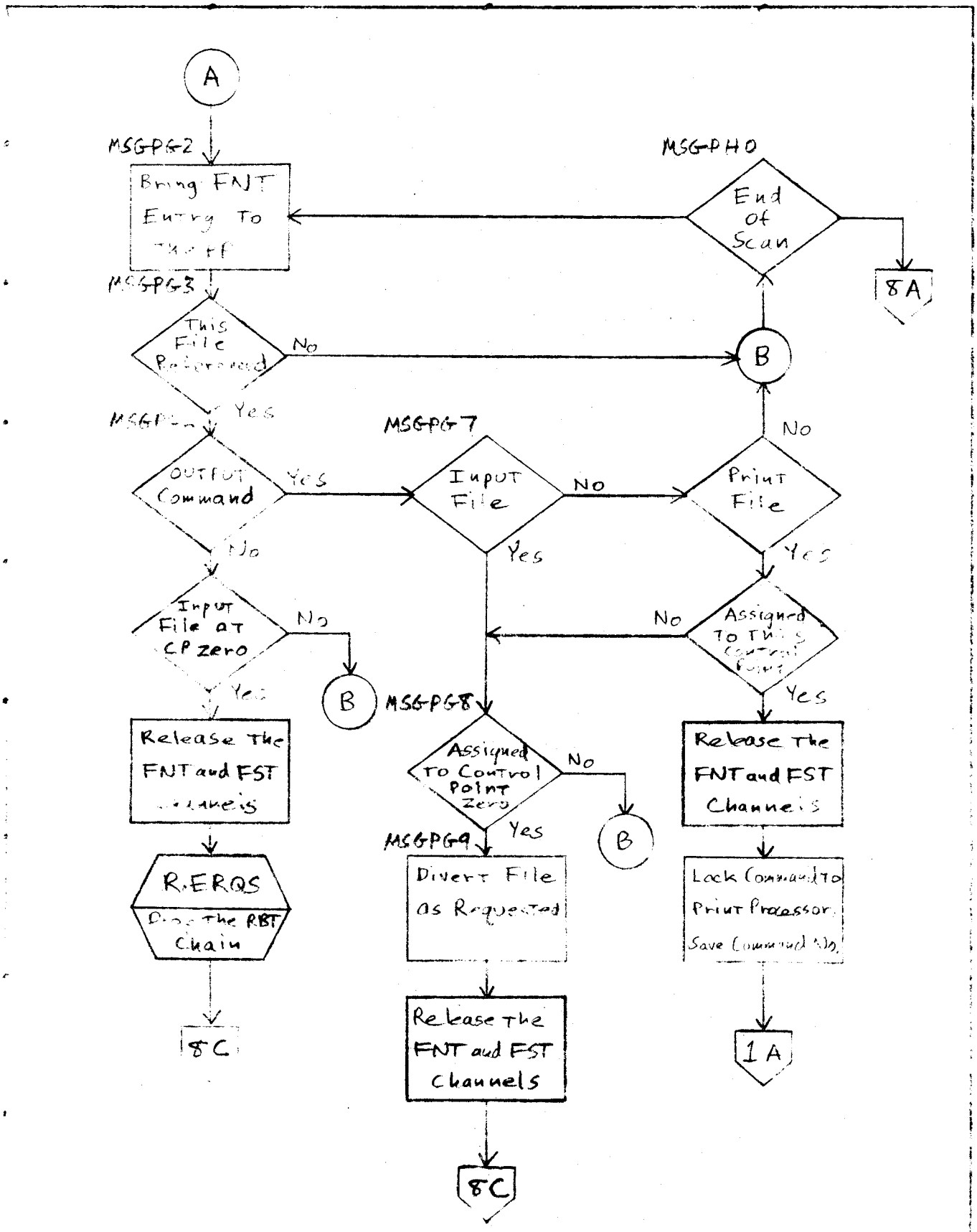

A
MSGPE0
YES
SAVE BREAK Address AND CURRENT POINTERS
2B
MSGPE1
NO
COMMAND Recognized, Set
QUIT COMMAND ?
YES
COMMAND STACK SET ?
YES
B-COMMAND ?
YES
NO
NO
NO
1A
H GEN
To GENERATE H Display
1B, 8B
8B
B
BGEN
Generate the B display
1B, 8B
8B
MSGPE2
H COMMAND ?
YES
Set COMMAND STACK TO H display
HGEN
GENERATE the H display
8.1A
1B, 8B
NO
MSGPE3
B COMMAND ?
YES
Set Command STACK TO B display
BGEN
GENERATE THE B display
8.1A
1B 8B
NO
4.1 A

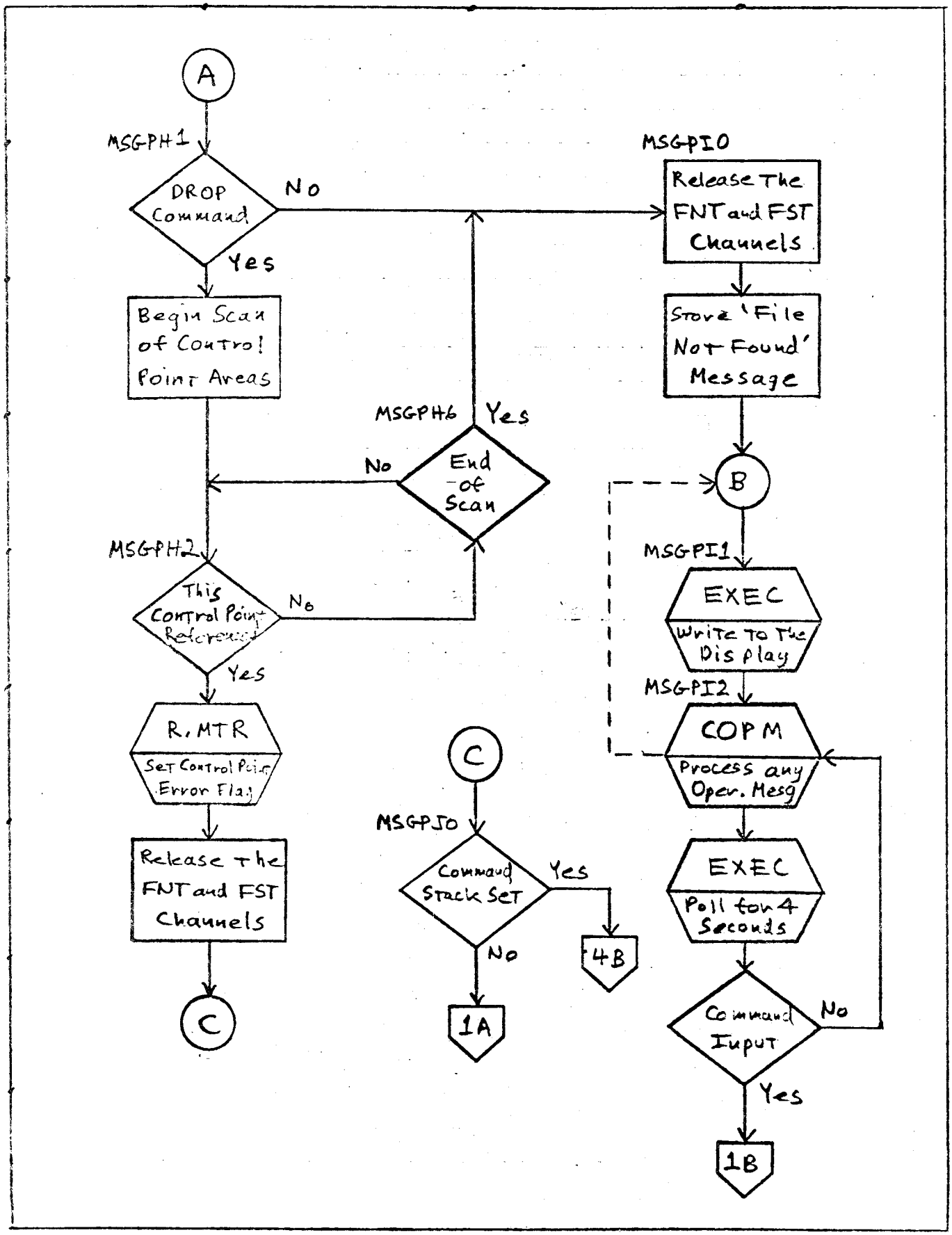

A
MSGPE4
TEST LOGIN OR LOGOUT
YES
MSGPE4A
Request Live Logout Save Command Number
1A
NO
MSGPE5
GO COMMAND
YES
Reset Command Lock, Save Command Number
1A
NO
MESSAGE COMMAND ?
YES
5A
NO
6A

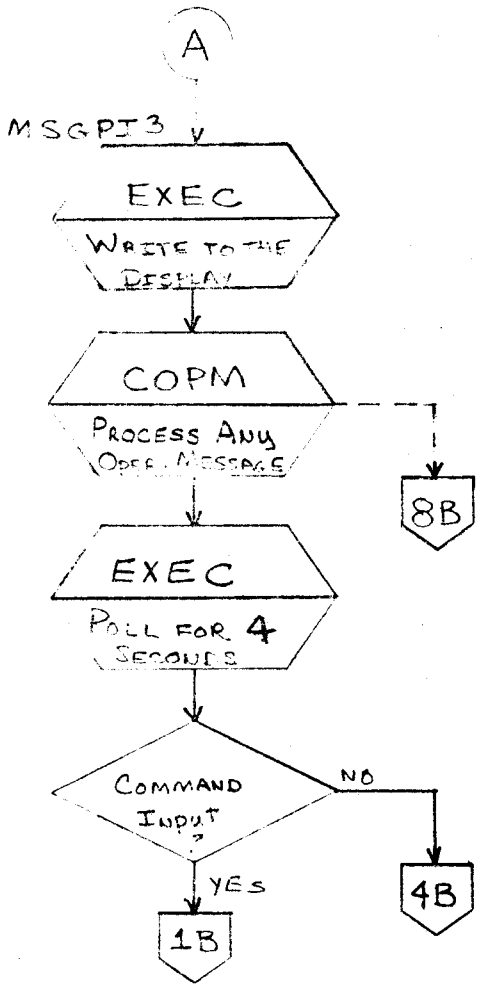

A
MSGPF0
Request Central Display
EXEC
Write Message Back to Display
MSGPF1
Display Assigned
No
Yes
MSGPF2
Display Area Clear
No
Yes
Write Message To Central Display
Clear Central Display Request
1A
MSGPF3
COPM
Process any Oper. Mesg.
EXEC
Poll for 2 Seconds
8B
No
Command Input
Yes
Clear Central Display Request
1B

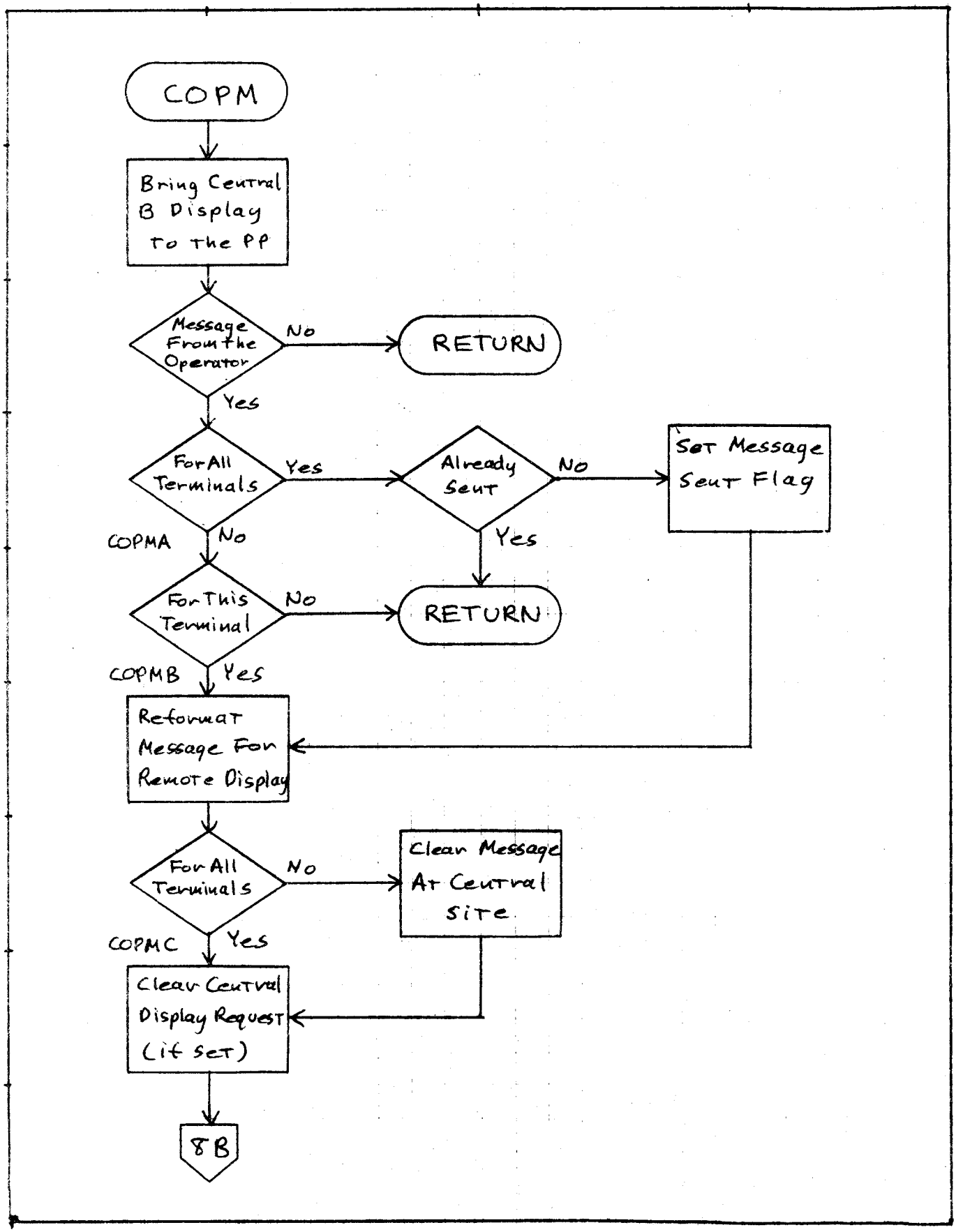
Page 6

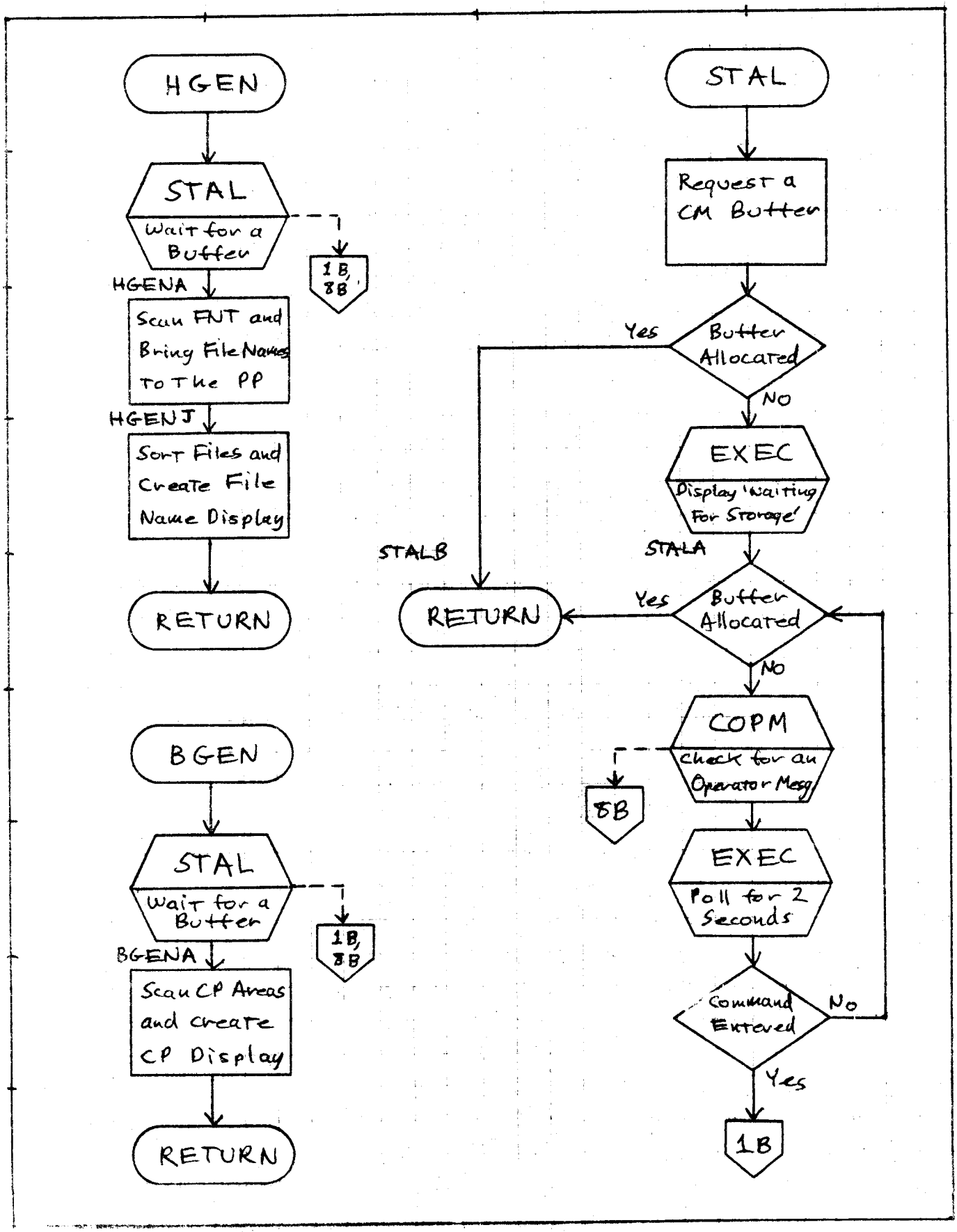

A
MSGP=0
DROP or OUTPUT Command
No
Yes
MSGP=1
Save Command Number
1A
R.RCH
R.RCH
Begin Scan of ENT
7A


A
MSGPG2
Bring FNT Entry To
MSGPG3
This File Retrieved
No
Yes
MSGPG
OUTPUT Command
Yes
No
Input File at CP zero
No
Yes
B
Release the FNT and FST Channels
R.ERQS
Drop the RBT Chain
8C
MSGPG7
Input File
No
Yes
MSGPG8
Assigned To Control Point Zero
No
Yes
B
MSGPG9
Divert File as Requested
Release the FNT and FST Channels
8C
MSGPH0
End of Scan
8A
B
No
Print File
Yes
No
Assigned To This Control Point
Yes
Release the FNT and FST Channels
Lock Command To Print Processor
Save Command Adr
1A


A
MSGPH1
DROP Command
No
Yes
Begin Scan of Control Point Areas
MSGPH6
End of Scan
Yes
No
MSGPH2
This Control Point Referenced
No
Yes
R.MTR
Set Control Point Error Flag
Release the FNT and FST Channels
C
C
MSGPI0
Command Stack Set
Yes
No
4B
1A
MSGPI0
Release The FNT and FST Channels
Store 'File Not Found' Message
B
MSGPI1
EXEC
Write To The Display
MSGPI2
COPM
Process any Oper. Mesg
EXEC
Poll for 4 Seconds
Command Input
No
Yes
1B


A
MSGPI3
EXEC
WRITE TO THE DISPLAY
COPM
PROCESS ANY OPER. MESSAGE
8B
EXEC
POLL FOR 4 SECONDS
COMMAND INPUT ?
NO
4B
YES
1B


COPM
Bring Central B Display To the PP
Message From the Operator
No
RETURN
Yes
For All Terminals
Yes
Already Sent
No
Set Message Sent Flag
COPMA
No
Yes
For This Terminal
No
RETURN
COPMB
Yes
Reformat Message For Remote Display
For All Terminals
No
Clear Message At Central Site
COPMC
Yes
Clear Central Display Request (if set)
8B


HGEN
STAL
Wait for a Buffer
1B, 8B
HGENA
Scan FNT and Bring File Names To The PP
HGENJ
Sort Files and Create File Name Display
RETURN
BGEN
STAL
Wait for a Buffer
1B, 8B
BGENA
Scan CP Areas and create CP Display
RETURN
STAL
Request a CM Buffer
Buffer Allocated
Yes
NO
EXEC
Display 'Waiting For Storage'
STALA
STALB
RETURN
Yes
Buffer Allocated
No
COPM
Check for an Operator Msg
8B
EXEC
Poll for 2 Seconds
Command Entered
No
Yes
1B

2.6.4 Message Processor Save Area and CM Buffer

Save Area: SAV1-SAV5, direct cells 63B-67B

SAV1 - This cell is used to save the return address for HGEN and BGEN (or any routine calling STAL).

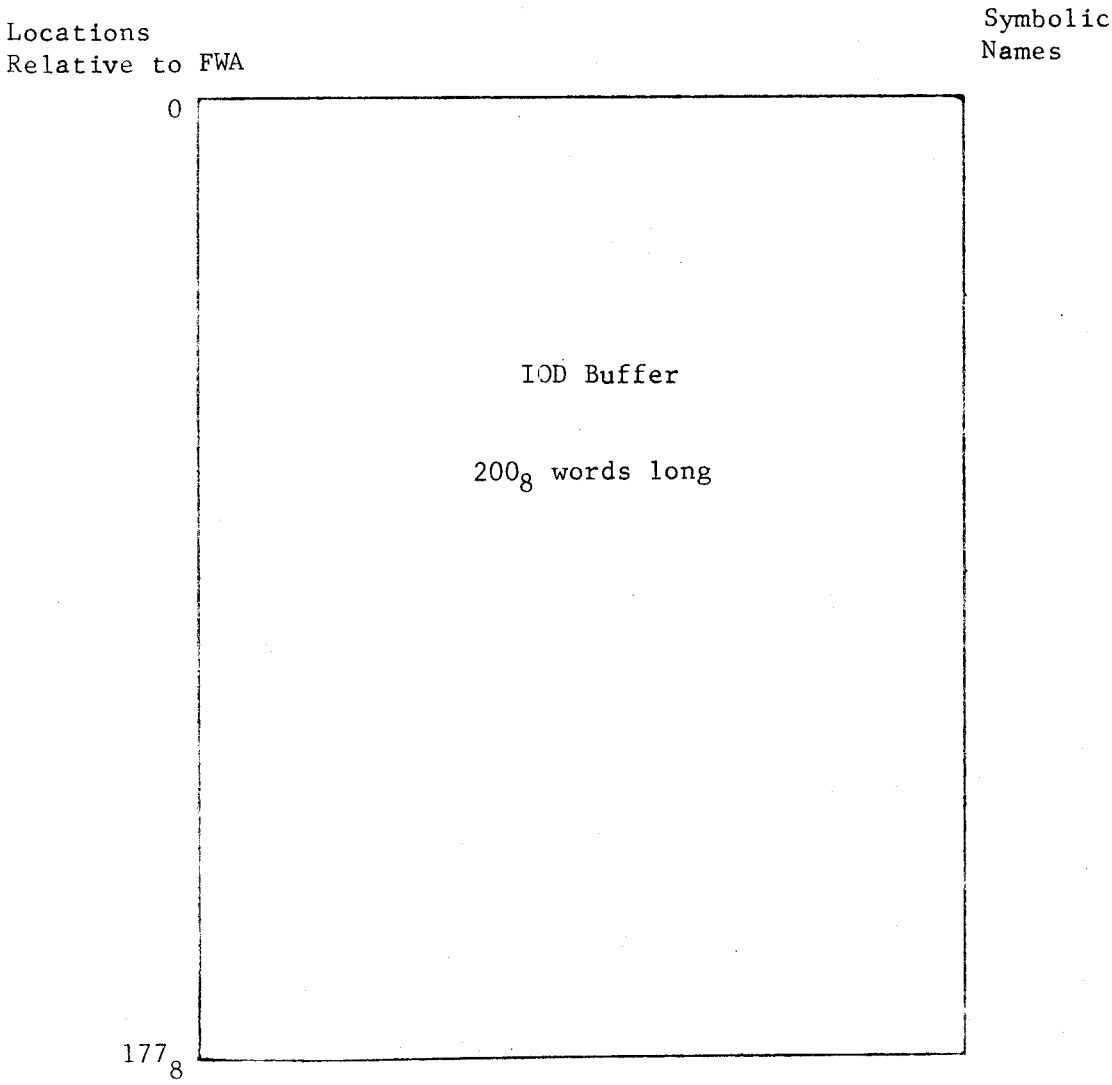
SAV2 - The low order bit of SAV2 is used to select the reader processor or the printer processor as the recipient of a LOGOUT command. This bit is toggled on successive LOGOUT commands to facilitate line flushing.

SAV3 - This cell is the command stack. It is set to zero when 4MU is activated and set non-zero if an H or B display is requested.

SAV4 - Not used.

SAV5 - This cell is used to save the return address for STAL.

CM Buffer: First word address = FWA = BAD*100_8+RA
200_8 words long



2.7 Memory Allocation and EXPORT Initialization

2.7.7 Narrative of EI2 and ALC Routines

PROGRAM - EI2
ROUTINE - STRT

STRT is the initialization program for EXPORT/IMPORT 200. STRT (EI2) is called by SCOPE when the central site operator enters "n.X EI2." into DSD. STRT sets the control point job name to EXPORT2, writes a control card (to call the allocator) into the control card buffer, and then sets the time limit and job priority. Finally, STRT requests a field length of $11{,}000_8$, and then overlays the PP with 1AJ which will load the allocator.

PROGRAM - ALC
ROUTINE - START

START is the initialization section of the allocator. START relocates the allocator to location 2, initializes all the system's CM tables, and calls IOD. START then sets the required field length to its minimum value and transfers to the allocator's automatic recall section.

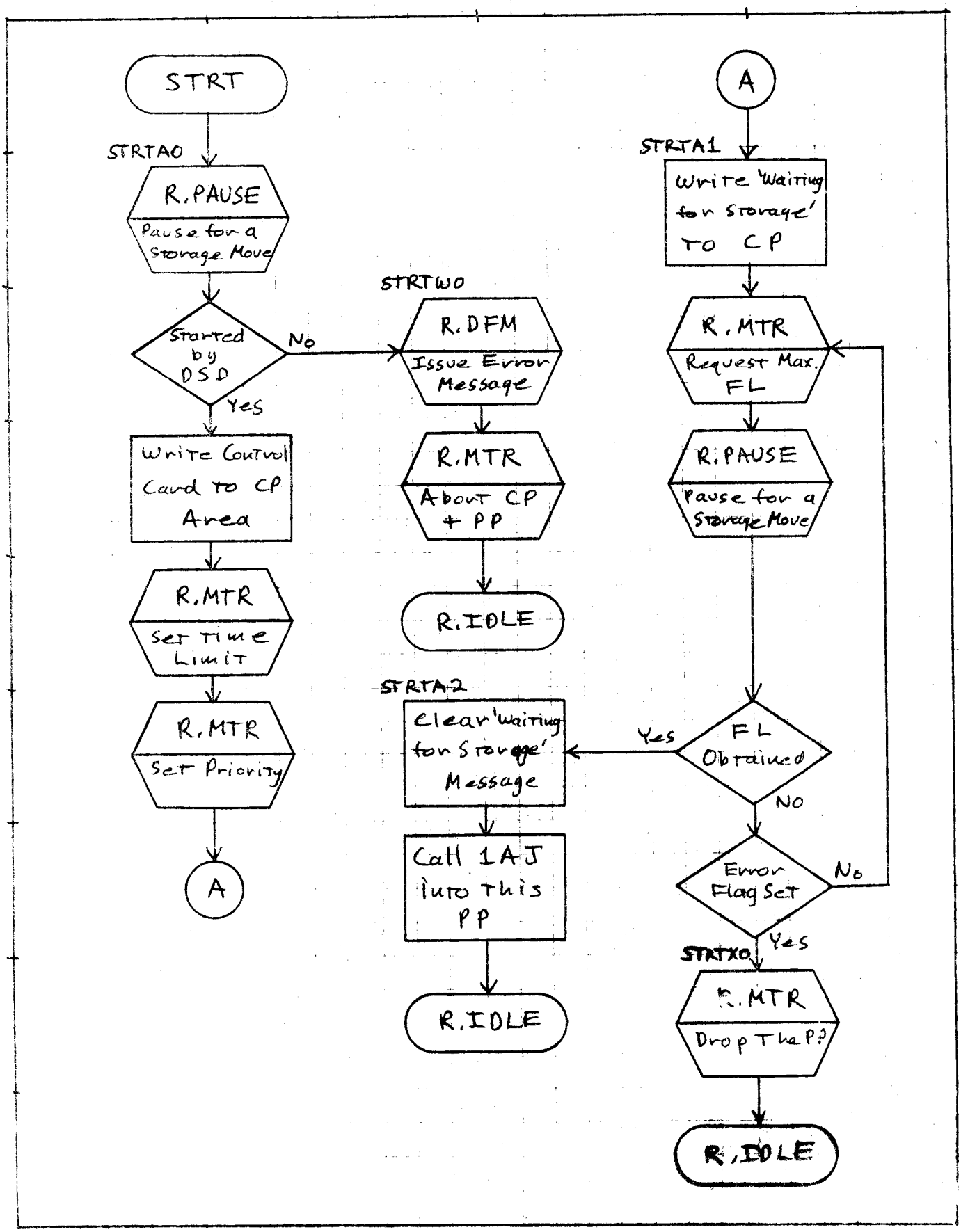
PROGRAM - ALC
ROUTINE - ALC

ALC is the E/I 200 CM allocator. Normally, ALC is in automatic recall waiting to be activated by ISD when a subsystem requests some storage allocate or release action. When activated, ALC scans the subsystem table, honors all release requests by clearing the assigned bits in the block table, and saves all allocation requests in the allocate stack. The block table contains 1 bit for each 100_8 CM word block that can be allocated, and the allocate stack is a stack of the storage requests sorted so that the largest request is first. After the subsystem table scan, ALC scans the block table to find a contiguous string of free {unallocated} blocks. When a string of free blocks is found, ALC scans the allocate table, and proceeds to allocate the blocks starting with the largest request first. When a request has been granted, the request is deleted from the allocate stack, and the stack is pushed up over the deleted entry. After all possible allocation has been accomplished on a string of free blocks, ALC proceeds to continue its scan of the block table until the next string of free blocks is encountered. The above procedure is repeated until the scan reaches the current field length. At this point, ALC recomputes the required field length by subtracting any free blocks

PROGRAM - ALC
ROUTINE - ALC {Continued}

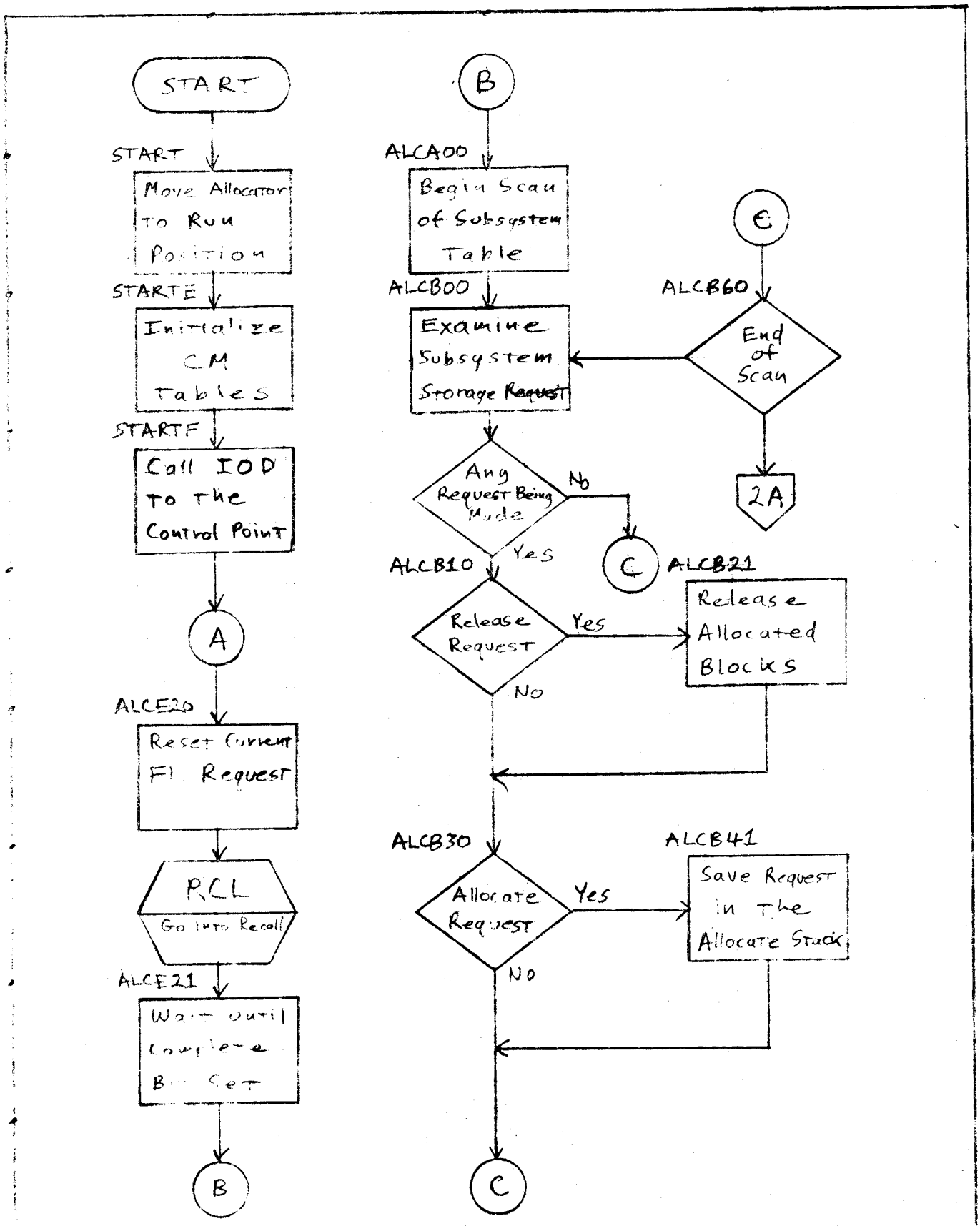
at the end of the current field, and then adding the number of blocks specified by unfilled requests in the allocate stack. This value is stored for lSD, {which will request the new FL} and then ALC goes into recall until reactivated by lSD.

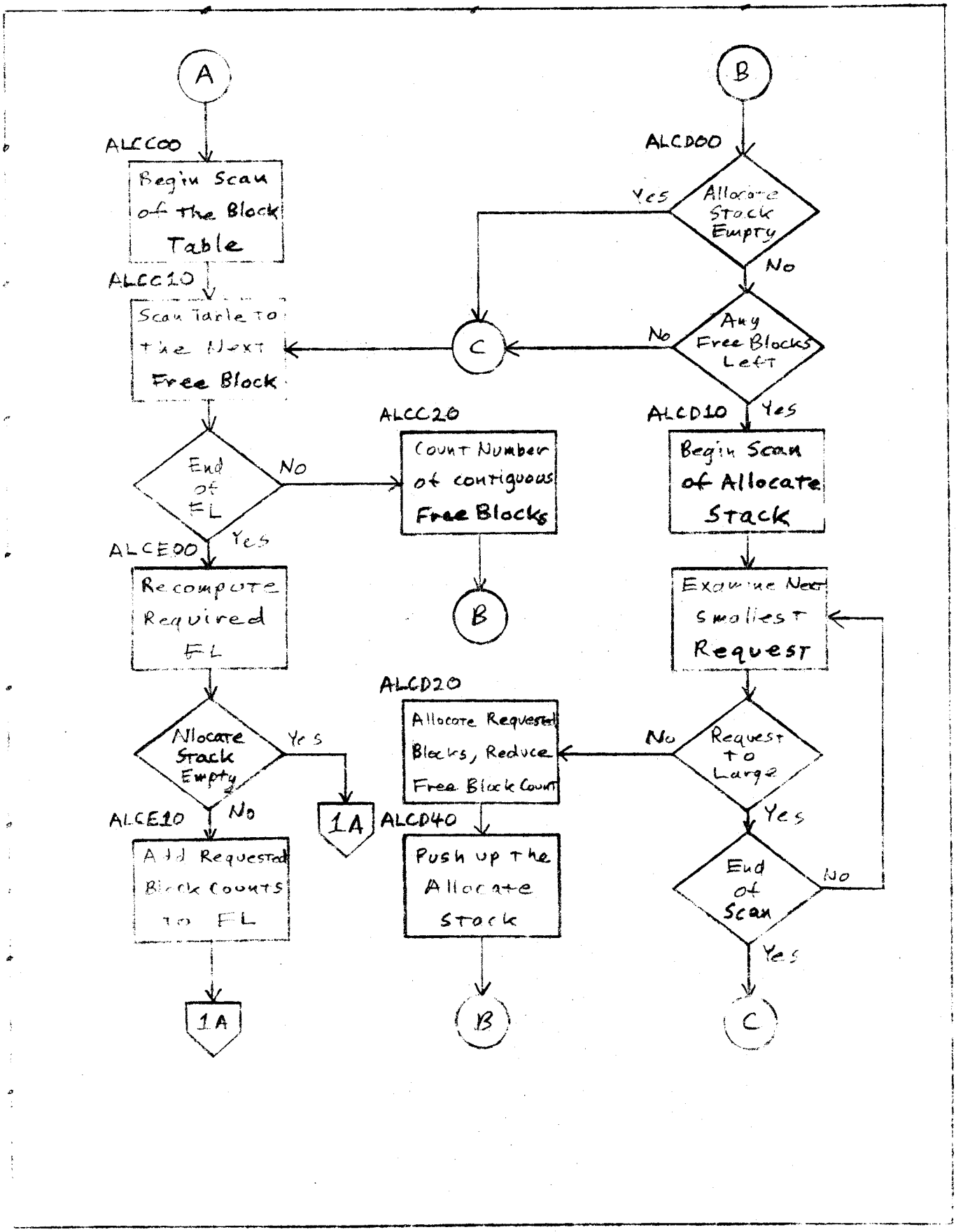
2.7.2 Flow Charts of EI2 and ALC Routines


STRT
STRTA0
R.PAUSE
Pause for a Storage Move
Started by DSD
No
Yes
Write Control Card to CP Area
R.MTR
Set Time Limit
R.MTR
Set Priority
A
STRTW0
R.DFM
Issue Error Message
R.MTR
About CP + PP
R.IDLE
STRTA2
Clear 'Waiting for Storage' Message
Call 1AJ into this PP
R.IDLE
A
STRTA1
Write 'Waiting for storage' TO CP
R.MTR
Request Max. FL
R.PAUSE
Pause for a Storage Move
FL Obtained
Yes
No
Error Flag Set
No
Yes
STRTX0
R.MTR
Drop The P?
R.IDLE

ALC

Page 1


START
START
Move Allocator to Run Position
STARTE
Initialize CM Tables
STARTF
Call IOD to the Control Point
A
ALCE20
Reset Current FI Request
RCL
Go into Recall
ALCE21
Wait until Complete Bit Set
B
B
ALCA00
Begin Scan of Subsystem Table
ALCB00
Examine Subsystem Storage Request
Any Request Being Made
No
Yes
C
ALCB10
Release Request
Yes
No
ALCB21
Release Allocated Blocks
ALCB30
Allocate Request
Yes
No
ALCB41
Save Request in the Allocate Stack
C
C
ALCB60
End of Scan
2A


A
ALCC00
Begin Scan of the Block Table
ALCC10
Scan Table to the Next Free Block
End of FL
No
Yes
ALCC20
Count Number of contiguous Free Blocks
B
ALCE00
Recompute Required FL
Allocate Stack Empty
Yes
No
1A
ALCE10
Add Requested Block Counts to FL
1A
B
Yes
Allocate Stack Empty
No
C
Any Free Blocks Left
No
Yes
ALCD00
ALCD10
Begin Scan of Allocate Stack
Examine Next smallest Request
Request to Large
No
Yes
ALCD20
Allocate Requested Blocks, Reduce Free Block Count
ALCD40
Push up the Allocate Stack
B
End of Scan
No
Yes
C

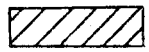
2.8 1JU Job Card Processor

1JU is loaded into a transient PP by 4RU when a job card needs to be processed. The location of the job card buffer (9 words) is in bytes 3 and 4 of 1JU's PP input register. The last word of the job card buffer is a communication word that is cleared by 4RU when 1JU is loaded. 4RU then goes into a WAIT loop until the first byte of the communication word becomes non-zero.

1JU loads and executes the system overlay 2TJ. If a job card error is detected by 2TJ, then 1JU terminates with byte 0 of the communication word set to 2. Otherwise, the job name (with the SCOPE sequence number supplied by 2TJ) and the time limit is written into the first word of the job card buffer and the job field length, priority, and ECS field length are written to bytes 1-3 of the communication word. In this case, byte 0 of the communication word is set to 1 to indicate that the job card was all right.

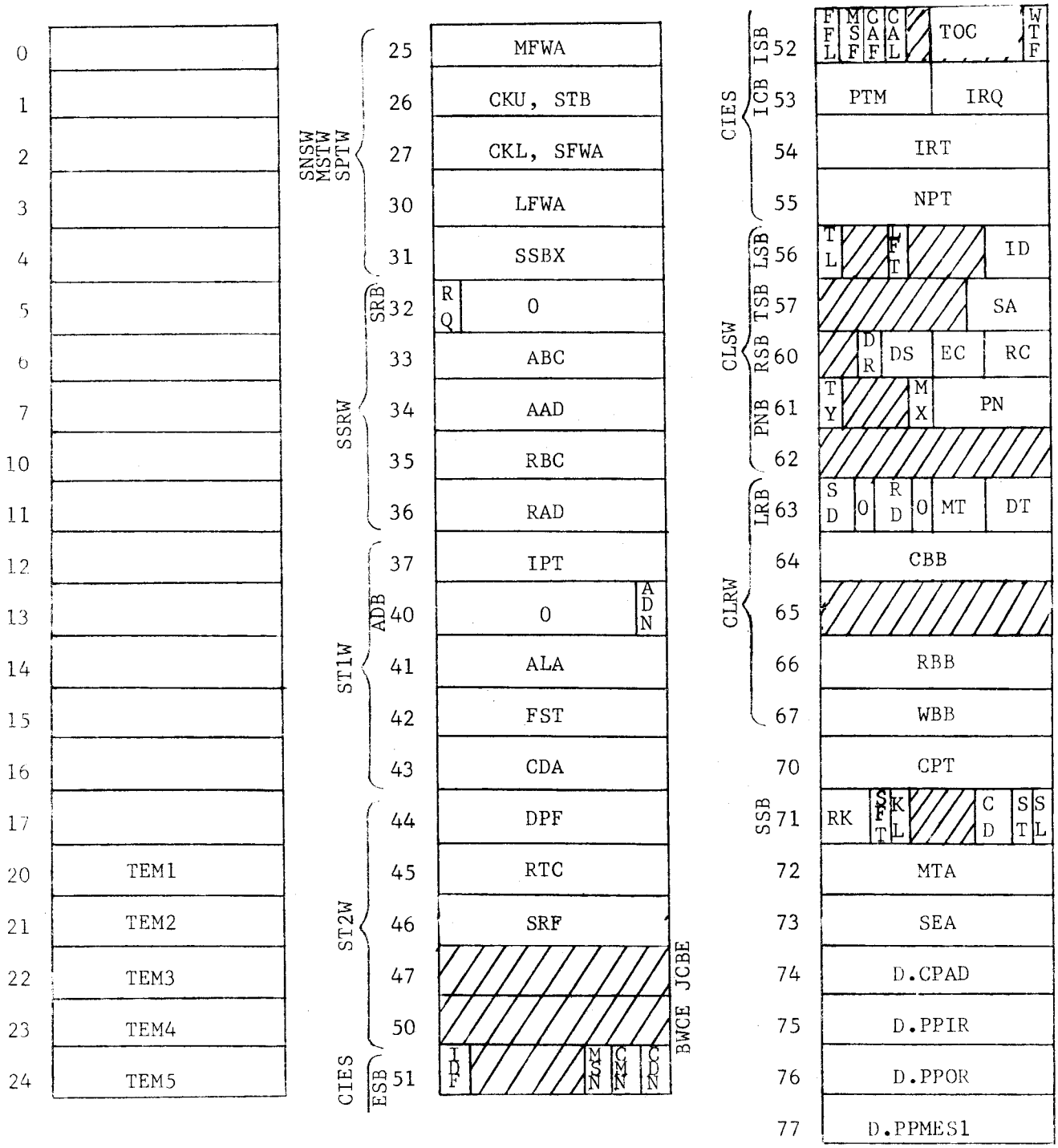
2.9 Direct Cell Assignments as Used by Each of the Overlays

The following tables show the bit names and bit designations for the flags and bits of the direct cells. Each level of the code has its own flags.

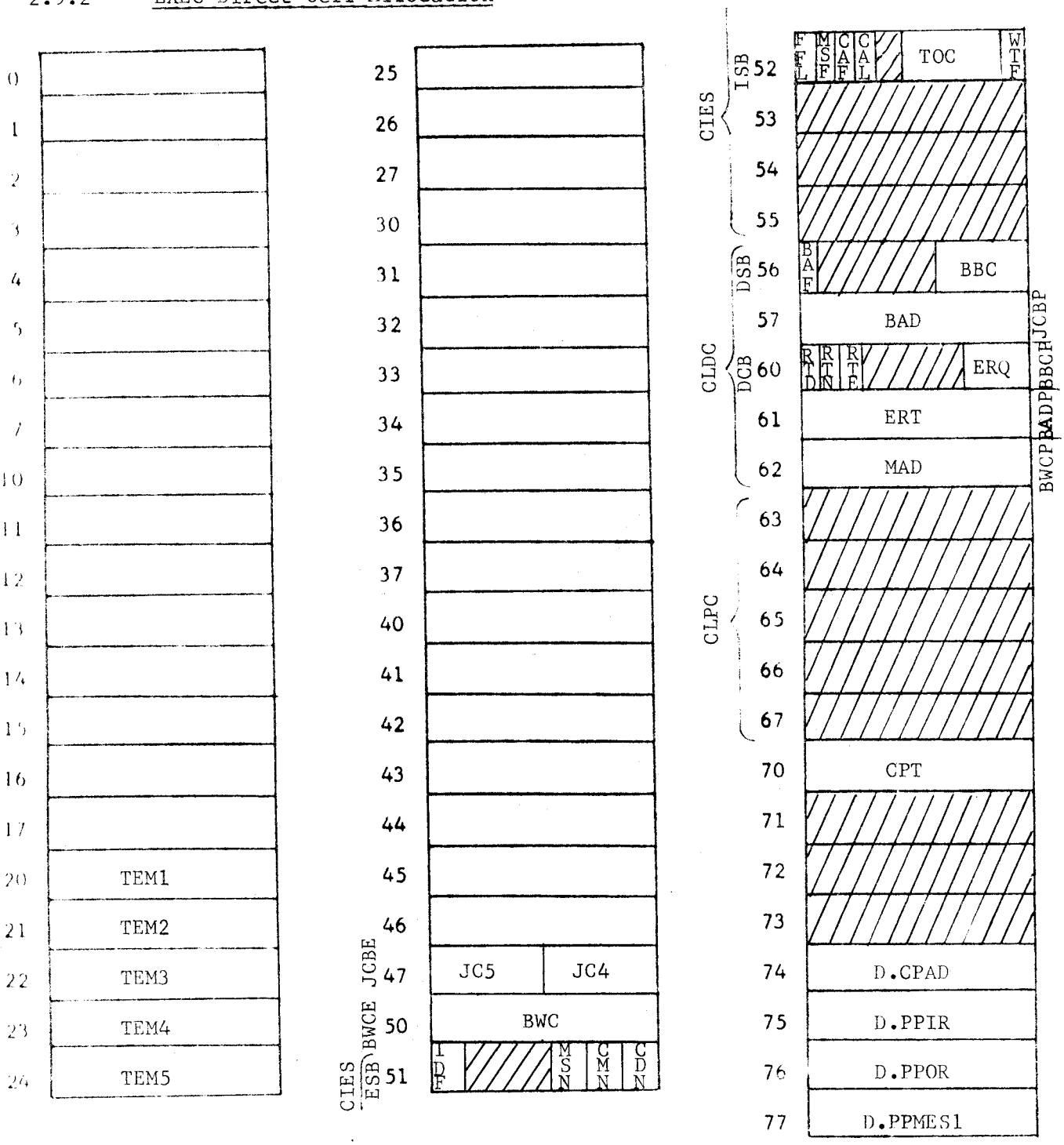
In the description which follows, cross-hatched areas  are preserved (neither set nor cleared) by the code using this series of cells. The areas labeled "0" are cleared. An area of five direct cells bracketed along the side is a central memory word, and its name or names appear outside the bracket. Byte names are inside the byte or beside the byte if the byte is further sub-divided by bit names. In general, the bit, byte and word designations were used in the code to aid in debugging and modification.

The following maps were useful in the development cycle during debugging.

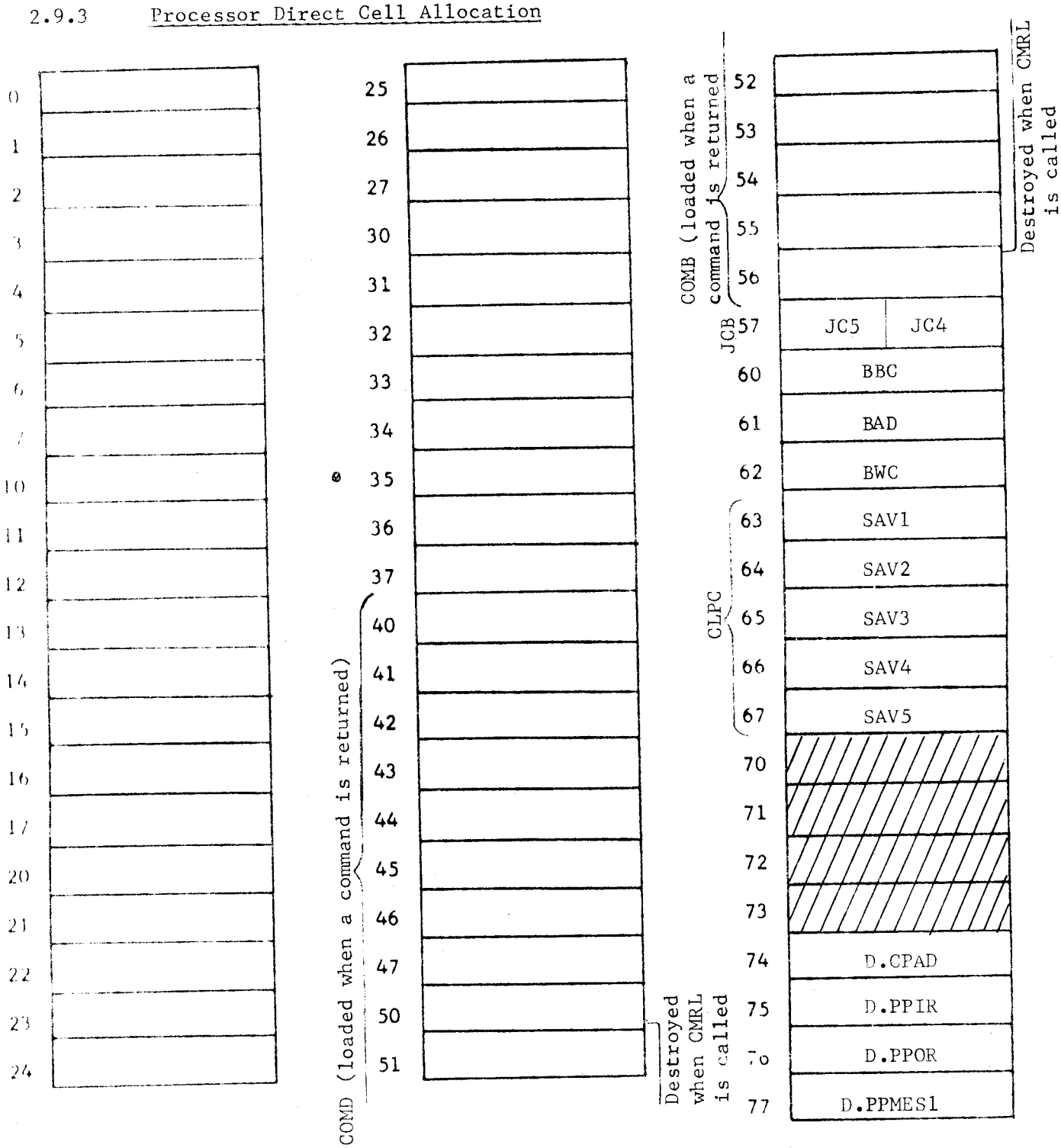
2.9.1 IODC Direct Cell Allocation



2.9.2 EXEC Direct Cell Allocation



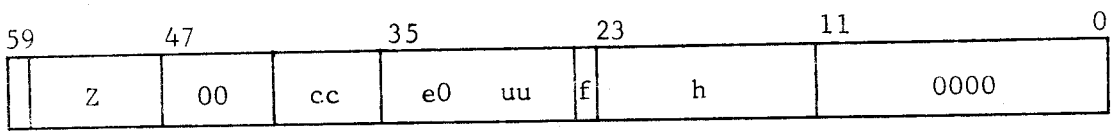
2.9.3 Processor Direct Cell Allocation



3.0 INSTALLATION INFORMATION

3.1 Installation EST Entry

Before the EXPORT/IMPORT system can be used, entries must be made to the Equipment Status Table (EST) for each multiplexor attached to the system. The format is the standard 6000 type non-allocatable:



Z = Status assigned by system (assembled as zero)

cc = channel number to which the 6671 is attached

e = equipment number of 6671 (set by switches on 6671)

uu = ordinal to a multiplexor descriptor table assembled into the I/O driver (Multiplexor Index)

f = on/off switch (assembled as zero)

h = hardware mnemonic in display code. For a 6671 this is DC or 0403_8.

Incidentally, the 6676 multiplexor which is driven by the same driver, but communicates only with teletypes has identical format with the hardware mnemonic YC or 3103_8.

The uu parameter is the ordinal to an entry in the multiplexor description table which is assembled into the initialization section of the driver. This table gives a detailed description of the configuration of devices connected to the multiplexor. It delineates which lines have teletypes and which lines have 200 USER terminals. The table also identifies which terminals have multiple 200 USER terminals in a

party line configuration, and what the site address of each terminal is. The exact configuration of this table is specified in Section 3.4.3.

This parameter is called Multiplexor Index, Multiplexor I.D. Number, and Multiplexor Number in other parts of this document.

The format for the EST macro used in assembling the EST is as follows:

TYPE	6000	channel, sync, unit, on/off
	TYPE	always DC for 6671
	Channel	channel to which 6671 is connected
	Sync	equipment number of 6671 (set by switches on 6671)
	Unit	multiplexor number as defined in the first parameter of the MPDEF macro
	on/off	on/off switch (on=0)

An example with the 6671 on channel ten with the synchronization switches set to one and the multiplexor number two would be:

```
DC    6000    10,1,2,0
```

3.2 Terminal Identification

Physically within each terminal there is a hardware method of differentiating among 16 terminals. This is accomplished by a site address code which is contained in each message from the terminal. The system will allow more than 16 terminals to be uniquely identified by combining a line number with the terminal supplied site address. Since we would like to allow switched line service such that users can call up on any of several lines, we combine a group of lines together as one unit, all with unique site address. Any of the 16 terminals which communicate on the group of lines can be uniquely identified. This means that a terminal may dial up on one line of the group to submit a job, and dial up on another to receive the output.

The user is given complete freedom at installation time to define which lines are associated together as a group.

Example

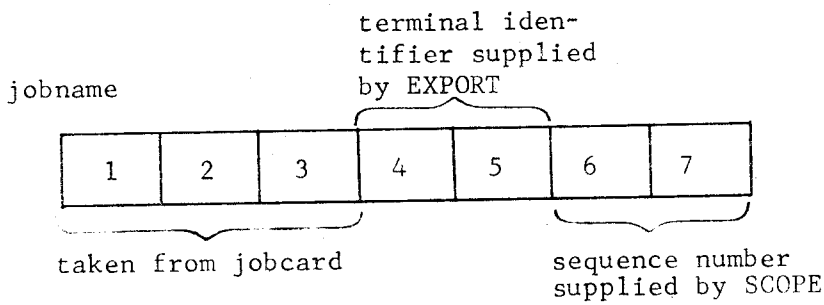
Suppose a configuration consisted of 22 terminals and one 6671 multiplexor. Assume that one terminal is connected by leased wire to one dedicated line, that two terminals are connected in a party line configuration to another dedicated line, and that all other lines are switched lines and have the ability to dial more than one multiplexor data set. The user can designate his dedicated lines as group A. The user must now spread his remaining 19 terminals across 14 multiplexor data sets. Since he can have only 16 terminals in one group, he must have two groups. He might divide his multiplexor in half such that 7

lines will be in each group. Let us call one group B and the other group C. He must also divide his 19 terminals such that each terminal may only dial into one group. He could assign 7 of the terminals to one 7-line group, thus assuring that these 7 terminals will always get a line; and the remaining 12 terminals to share the other 7-line group, giving them lower priority in access to the machine. This configuration is arbitrary. The user may decide to assign 9 terminals to one group of lines, and 10 terminals to the other group. Alternatively, he may decide to assign 3 lines to one group of 3 terminals, and let the remaining 16 terminals contend for the remaining 11 lines.

For the user to actually assign the group numbers, he must make entries into a line group number table before assembling the transient pp program of EXPORT. The user will be allowed to assign any line to any group individually. There can be as many groups as there are lines into the multiplexors. The procedure for constructing this table is defined in section 3.4.

3.3 Job Name Identification

The association of jobs and terminals is made by replacing two characters of the job name with an identifier which is unique to the terminal.



The fourth character is determined by the line group as described in the previous section. It is picked out of the line group table and is dependent upon the particular group of lines from which the job was read.

The fifth character is derived from the station address. The display coded characters U through Z and 0 through 9 will represent the site addresses of 160_8 through 177_8 respectively. It may be noted that the replaced fifth character of the job name will not be A through T for E/I 200. EXPORT/IMPORT 8130 and EXPORT/IMPORT 8231 can use these characters to identify jobs submitted through their respective terminals.

All remote jobs submitted through the EXPORT/IMPORT systems have a disposition code which is different from jobs submitted from the central site, and therefore there is no restriction on the use of job names for jobs submitted from the central site.

3.4.2 Installation Options

For the EXPORT/IMPORT 200 system to operate properly, a large number of parameters must be set at assembly time. A description of the parameter and its name follows. They are defined in the common deck IODCOM.

The following parameters must be set according to the particular site configuration.

M71 is the maximum number of 6671 multiplexors which will be run at one time. This must be less than or equal to 4. Note additional restrictions in description of M76.

M76 is the maximum number of 6676 multiplexors which will be run at one time. This must be less than or equal to 2. M76 must be such that M71+2*M76 is not greater than 4.

M is the number of multiplexors which are defined by the multiplexor port definition table and an entry in the EST. This must be less than or equal to 4, and will normally be equal to the total number of multiplexors available. However, a given multiplexor may be defined more than once in the multiplexor port definition table and EST by changing its multiplexor index. For example, if one 6671 is available with three CRT's on ports 0, 1, and 2, and two TTY's on ports 3 and 4, and if the user

finds it desirable to disconnect the TTY's for some runs, he may define two different multiplexor configurations as follows:

```
MPDEF  0,0,A
MPDEF  0,1,B
MPDEF  0,2,C
MPDEF  0,3
MPDEF  0,4
MPDEF  1,0,A
MPDEF  1,1,B
MPDEF  1,2,C
```

with the corresponding EST entries:

```
0000  00CC  e000 fo403  0000
0000  00CC  e001 fo403  0000
```

where f would be unlocked for only one entry at a given time by turning on or off the appropriate equipment. (See Section 3.4.3 for a description of MPDEF.) Using the entry with mux index = 0, the three CRT's and two TTY's would be serviced; the TTY's would be ignored by using the entry with mux index = 1.

CRT is the maximum number of 200 USER groups (ports). This is the number of ports on the multiplexors which are connected to 200 USER Terminals. It may be less than the number of 200 USER Terminals if there are party line configurations. It must not be greater than the smaller of 16*M71 or 32.

TRCT is the maximum number of 200 USER Terminals.

LPCRT is the largest 200 USER Terminal line number. This may be calculated from:

$$\sum_{M=0}^{MM-1} (\text{no. of ports})_M + (\text{largest 200 USER port no.})_{MM}$$

where MM = last (highest numbered multiplexor index) mux that has a 200 USER attached to it. If a configuration consisted of two 6671's with 16 ports of the first 6671 being used and port 4 (0-4) of the second 6671 was the largest numbered port with a 200 USER Terminal, then LPCRT should be 20.

TTY is the maximum number of teletypes. It must not be greater than the smaller of (128-TCRT) or (16*M71+64*M76-CRT).

S is the number of the highest numbered available subsystem +1. Note that subsystem zero is always available; hence, S must be at least 1. For example, if only EXPORT/IMPORT 200 is to be run (no SENTRY or RESPOND II), then S=2 (EXPORT/IMPORT 200 subsystem no.) +1=3.

For the EXPORT/IMPORT 200 system, the following parameters should be set:

MAXFL is the maximum number of 100_8 word blocks that might be allocated using the EXPORT memory allocator. If RESPOND is running, this parameter is not used since RESPOND performs the allocation.

CRBL is the number of 100_8 CM word blocks which will be assigned to a terminal when it is reading card data. The minimum is 3; the normal is 5.

LPBL is the number of 100_8 CM word blocks which will be assigned to a terminal when it is printing. The minimum is 3; the normal is 5.

The following parameters must be set for RESPOND:

RESPTR is the relative address from which the JUNGLE, CIRSTACK, and JULAE pointers are read. See RESPOND.

CIRSTAX is the number of entries in CIRSTACK.

The following parameters are required by the I/O Driver:

IMP is the EXPORT/IMPORT 200 availability. If EXPORT/IMPORT 200 is to be run, this must be set =1. Otherwise, set it to zero.

RES is the RESPOND II availability. If RESPOND II is to be run, this must be set =1. Otherwise, setting it to zero will cause it to be made unavailable and will also eliminate the assembly of some parts of the I/O Driver.

SEN is the SENTRY availability indicator. If diagnostics (SENTRY) are to be run, this must be set =1. Otherwise, setting it to zero will cause SENTRY to be made unavailable and will also eliminate the assembly of some parts of the I/O Driver.

JUN is the total number of PP buffers available. Since JX=11 buffers are always available by using PP Resident, JUN must be at least equal to JX. If more buffers are required, they must be made available in high PP memory. JUN-1 must not be less than (TTY+4)/5.

MTAR is the maximum turn around time between the completion of a write and the beginning of a response from a 200 UT. The value is in milliseconds and should be set to 250 or greater depending upon the longest expected transmission distance.

3.4.3 Multiplexor Port Definition Macro

MPDEF MUX,PRT,T,SA1,SA2,---,SA16

MPDEF is used to define each port to be used by IOD. In the above example, MUX is the multiplexor index ($0 \leq MUX \leq 3$), PRT is the port number ($0 \leq PRT \leq 15$ for 6671, $0 \leq PRT \leq 63$ for 6676), and the remaining parameters are optional. If T is missing, this is a teletype port. If T is any character from A to 9, this is a CRT port and the specified character is the first character of the Terminal ID. (Job character 4 for EXPORT.) If T is specified and SA1 is missing, this is not a party line port. If T and SA1 are specified, this is a party line port and SA1 is a site address for one of the connected terminals. SA2 → SA16 are the other site addresses. The first missing SA terminates the port definition.

MPDEF 0, 1

TTY on port 1 of multiplexor 0.

MPDEF 1, 6, A

CRT on port 6 of multiplexor 1. The first character of the terminal ID will be A.

MPDEF 1, 2, B, 0, 5, 13, 2

CRT's on port 2 of multiplexor 1. The first character of the Terminal ID will be B. The allowed site addresses are 0, 5, 13, and 2.

The following tables are assembled from information specified about the multiplexor ports by using the MPDEF macro.

PTABLE, LTABLE, and PTTABLE will contain one entry for each multiplexor defined. These entries are ordered by the multiplexor index.

MDTABLE will contain an entry for each terminal defined. Non-party line ports will have an entry for each port. Party line ports will have one entry for each site address specified. The MDTABLE entries are ordered by multiplexor index and then by port number.

PTABLE entries specify the number of ports being used on each multiplexor.

LTABLE entries specify the number of terminals connected to each multiplexor. This also is the number of MDTABLE entries for each multiplexor.

PTTABLE entries are the pointers to the first MDTABLE entry for each multiplexor.

MDTABLE entries are of the following format:

1	1	4	6
TY	PL	SA	PN

where: TY=1 for TTY, =0 for CRT

PL=1 for party line, =0 for non-party line

SA = site address for party lines

PN = port number

3.4.4 Subsystem Definition Macro

The macro system has been defined. The parameters required are:

1) A four-character identifier which is to be used to request the subsystem from the terminal.

2) The three-character name of a PP program to be loaded by the I/O Driver.

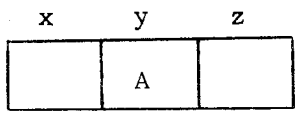
3) A work specifying the availability of the subsystem; 1= available, 0=not available, and if the subsystem requires an initial load, the KL (100B) bit should be set. For example, for EXPORT/IMPORT 200 the macro would be specified as:

```
SYSNM     IMP0,1EU,IMP+KL
```

4.1 Coding Conventions

The code in the transient PP overlay uses some conventions which associate labels of constants with a particular named field in a PP word. These constants are generated by a macro call and are generally used to define masks and shift constants for accessing the byte of the field. The following example shows a label "A" for a field in a PP word and the various associated constants.

Given a flag in a 12 bit field



x = 3
y = 4
z = 5

SHN	SL→A		SL→A = x + 6 = 9	high order bit of A is shifted to the A register sign bit.
LPC	MP→A	mask	MP→A = x0's, y1's, z0's = 0740_8	
LPC	-MP→A	zero	-M→A = x1's, y0's, z1's = 7037_8	
LDC	5*PS→A		PS→A = 2^z = 0040_8	

The macro DFN will automatically define these values.

A	DFN	x, y

Other Conventions:

D.RA
D.FL } are not used by the program

All references to CM go through a routine called CTFL, which calls R.TFL to check address and add the RA. R.TFL has its own locations in PP resident.

4.2 Character Set of the Terminal

The % becomes a ¬ (logical not) in the 6000. It is printed as a % at the terminal, but is interpreted as a ¬ in the 6000 for compilation, central site printing, etc. The ; (semicolon) is supported as is in both systems. The carriage return is translated into a two-character internal code.

4.4 Bringing Up of Central Site Software

When the central operator types n.X EI2, the system display package DSD calls EI2 to a peripheral processor at the control point n, named in the request. EI2 sets the job name = EXPORT2, sets priority = 7700, sets time limit = 10000, and sets control cards to call ALC into the system. It then overlays its PP with 1AJ. 1AJ examines the control card area and calls ALC into execution. ALC does some small initialization and reorigins itself into location 2 (from location 102 as set by the loader) to save space, and then calls IOD. IOD accesses the EST to get a multiplexor, references its own assembled table of the description of this multiplexor, and finishes initializing central memory tables. It then calls 1EU, which is the basic controlling overlay of the EXPORT system. 1EU is called whenever a line needs processing by EXPORT (needs more print data, for example), or when a time limit of a few seconds has been exceeded (4 seconds when no lines are active, 1 second when lines are active but not requiring storage, ¼ second when storage is being allocated).

1EU is transient, while IOD is dedicated. Both are PP programs. ALC is a central program and it remains resident.

4.5 EXPORT/IMPORT 200 File Definitions

The following definitions apply to files within the SCOPE 3.0 system that are tagged as belonging to EXPORT/IMPORT 200:

a. "An E/I 200 File" is a file that 1) has the name wwwxyzz where www are any alphanumeric characters, xy is the Terminal ID of the terminal associated with the file, y is any character between U and 9, zz is a SCOPE supplied sequence number, and 2) has a disposition code of $4XXX_8$.

b. "An E/I 200 Output File" is an unlocked E/I 200 file of type output or local with a non-zero priority.

c. "An E/I 200 Print File" is an E/I 200 Output file whose disposition code is equal to $404X_8$.

d. "An E/I 200 Input File" is an unlocked E/I 200 file of type input with a non-zero priority.

EXPORT/IMPORT 200 output files that are assigned to control point zero and are not print files are diverted to the central site by clearing the high order bit of the disposition code. EXPORT/IMPORT 200 input files are given an initial disposition code of 4000_8 before being assigned to control point zero.

INPUT/OUTPUT DRIVER DOCUMENTATION

Part I - EXTERNAL INTERFACE SPECIFICATION

Part II - INTERNAL SPECIFICATION

Part I - EXTERNAL INTERFACE SPECIFICATION

1.0 INTRODUCTION

The EXPORT/IMPORT 200 system uses a dedicated peripheral processor (PP) to transfer data between the multiplexor and central memory. This PP program is called the Input/Output Driver (IOD). The same program (IOD) is used in conjunction with RESPOND and SENTRY to perform input and output through the same communications facilities. The following sections document how various subsystems (EXPORT/IMPORT 200, RESPOND, SENTRY) communicate with IOD.

The following list shows the names of various tables and pointers used to interface the Driver and Subsystems. The tables and pointers have been given letters so that they may be easily referenced.

CENTRAL MEMORY STORAGE MAP FOR IOD

Reference Letter	Description	Size (CM $words_{10}$)
A	CM Buffer Allocator Status Word	1
B	System Pointers	1
C	Equipment Table	M
D	Multiplexor Status Table	(M+LP)/5+2
E	Subsystem Tables	8(S+1)
F	Line Tables	12(T)
H	Working Buffers	X

Key:

M = the maximum number of multiplexors

S = the maximum number of subsystems

LP = the maximum number of lines (ports) for all multiplexors

T = the maximum number of terminals for all multiplexors

X = the total number of words requested by all subsystems for working buffers (a multiple of 64)

Notes:

The location of A will be in the low order byte of IOD's input register.

The location of B is = A+1, and will be supplied with each subsystem call.

B contains the pointers to D, E, F.

C = B+1

B through F are contiguous and have relative address (to RA) of 12 bits or less.

B contains the lengths of E and F.

D contains the lengths of D.

F contains the pointers to H.

H starts at a 100_8 CM word boundary.

Values M, S, LP, T and the maximum value of X are assembly time parameters.

$1 \leq M \leq 4$

$1 \leq S \leq 7$

$1 \leq LP \leq 128$

$1 \leq T \leq 128$

2.0 CENTRAL MEMORY STORAGE MAP

3.0 SYSTEM POINTERS - SYSPTR (B)

12	3	9	12	12	12
MFWA	S	T	SFWA	LFWA	

MFWA : FWA of the Multiplexor Status Table (D)

S : total number of subsystems; 8(S+1) is the length of the Subsystem Tables (E)

T : total number of terminals (lines) connected to all multiplexors; 12(T) is the length of the Line Table (F)

SFWA : FWA of the Subsystem Tables (E)

LFWA : FWA of the Line Table (F)

When IOD is loaded, the 5th byte in the input register (call word) is the location SYSPTR-1. IOD will assume that SYSPTR has been initialized. The middle byte in the input register contains flags indicating the availability of the 7 possible subsystems.

5	7
	AR

The 2^{7-SS} bit is 1 when subsystem SS is available. ($1 \leq SS \leq 7$)

4.0 EQUIPMENT TABLE (C)

Each table entry (i.e., 1 CM word) represents one multiplexor. The table is ordered so that the 6671 multiplexors appear before 6676 multiplexors. The format of each entry is as follows:

3	2	1	2	4	12	12	12	12
ES	.	MU		CH	PA	PB	LA	LB

ES : equipment select code $0 \leq ES \leq 7$

MU : 0=6671, 1=6676

CH : channel number of the multiplexor

PA : multiplexor port number of first line

PB : multiplexor port number of last line

LA : line number of first terminal on this mux

LB : line number of last terminal on this mux

The location of each entry in the equipment table is determined by the multiplexor index from the EST entry. The maximum number of multiplexors (M) must be large enough to accommodate the largest EST multiplexor index. The table entry will be zero if the multiplexor is currently not in use.

5.0 MULTIPLEXOR STATUS TABLE (D)

This table is of variable length (M+LP)/5+2 depending upon the number of multiplexors (M) and the total number of ports (LP). It is written by IOD each multiplexor cycle and contains all values which change that frequently. The first CM word contains items of general interest while the second and subsequent words contain hardware status information for diagnostic purposes.

5.1 First Word of General Information

12	24	12	12
WC	CK	MM	LPP

WC : number of CM words in this table = (M+LP)/5+2

CK : millisecond clock (24 bits)

MM : the actual number of multiplexors in use

LPP: the actual number of lines (ports) currently being used for all multiplexors

5.2 Second and Subsequent Words

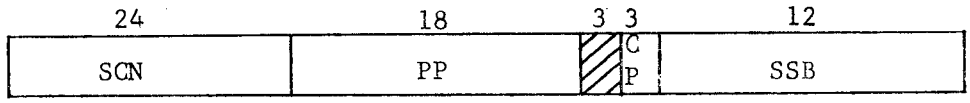
These words contain status and input words from the multiplexor. Each item is 12 bits long with 5 items packed per CM word. The first M items are the multiplexor status words. The next LPP items are the 12 bit input word for each line.

Item	Description
MS1 ... MSM	12 bit status word for each multiplexor $1 \leq M \leq 4$
LSO ... LSX	12 bit input word for each line $X = LPP-1$, $1 \leq LPP \leq 128$

6.0 SUBSYSTEM TABLES (E)

The subsystem table has an 8 CM word entry for each subsystem which is made up of three parts as follows:

6.1 Subsystem Name and Status Word

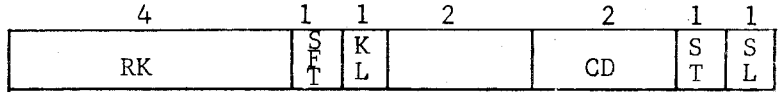


SCN: the first 4 characters of the subsystem call name* in Display code. SCN is used by subsystem zero (1ID) to assign idle lines (i.e., those assigned to subsystem zero) to other subsystems. When SCN compares with the first 4 characters of a requested subsystem name, subsystem is assigned. SCN is initialized by IOD and is never altered.

PP : the 18 bit name of the main PP program which represents the subsystem. PP is used by 1SD (IOD's auxiliary transient PP program) when the subsystem is to be loaded. PP is initialized by IOD and is never altered.

CP : the control point number to which IOD (and thus the subsystem) is assigned. CP is initialized by IOD and is never altered.

SSB: the subsystem status byte used to control loading and describe the status of the subsystem. SSB is altered by both 1SD and the subsystem.



*LOGIN (RESPOND)
IMPORT (E/I 200)
TEST (SENTRY)

RK : Automatic Recall Time (see KL below). Not altered by 1SD.

SFT : Set to 1 by 1SD when LFT is set in the LSB for some line assigned to this subsystem.

KL : Timed Reload Flag. When the subsystem sets KL=1, IOD will reload the subsystem automatically (regardless of whether any line is ready) when RK=CK/200_8. When the subsystem is loaded, KL=0 indicates that the loading was not due to CK/200_8 being equal to RK. If, however, KL=1 after loading, then CK/200_8=RK.

CD : Central Display Status (first line of the control point day file area) (used for issuing messages to the operator). Set to 0 by the subsystem when the display is not needed. Set to 1 by the subsystem when the display is required. Set to 2 by 1SD after being set to 1 by the subsystem. When CD=2, the display is assigned to the subsystem and is locked out to other subsystems. The subsystem must set CD=0 to release the assignment. Once CD is set to 2, the subsystem must also wait until the first byte of the message area is set to zero (by the operator or another subsystem) before

writing any message out. (Messages coming from the operator have a 2 character prefix to identify the subsystem and do not require assignment of the display. It is the responsibility of the subsystem to periodically check for this input and clear the first byte when the message is read.)

ST : Subsystem Status. When ST=0, the subsystem is not loaded; when ST=1, the subsystem is loaded.

SL : Status Word Lock. When SL=0, the subsystem may write the word; when SL=1, 1SD may write the word.

The interaction of 1SD and the subsystem is as follows:

ST=0, SL=1: The subsystem is not loaded, and therefore may not alter SSB.

If: RK=CK/200_8 and KL=1 or

SFT is set or

CD was 1 and is set to 2 or

TL is set in some line assigned to the subsystem

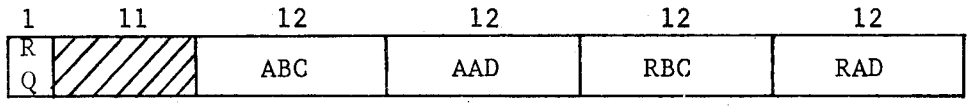
Then: 1SD will set ST=1, SL=0 (also KL=0 if RK≠CK/200_8 and KL=1 previously), and then will load the subsystem.

ST=1, SL=0: The subsystem is loaded and may set or reset RK, SFT, KL, CD, or ST until it sets SL=1. 1SD will not alter SSB while SL=0.

ST=1, SL=1: The subsystem is loaded but may not alter SSB. 1SD will act exactly as if ST=0, SL=1, except that when SL is set to zero, the subsystem will not be loaded.

ST=0, SL=0: The subsystem is not available. 1SD will ignore SSB.

6.2 Buffer Request Word



RQ : 1 = action requested by the subsystem.

0 = request has been attempted and possibly completed.

ABC: the number of consecutive blocks of 100_8 CM words in a buffer to be allocated.

AAD: the FWA/100_8 of the allocated buffer.

RBC: the number of consecutive blocks of 100_8 CM words in a buffer to be released.

RAD: the FWA/100_8 of the buffer to be released.

A CP memory allocation program (provided by RESPOND when loaded or by EXPORT when RESPOND is not loaded) provides the buffer management. The allocator is fired up periodically by IOD to service the buffer requests.

The RQ bit is used as an interlock between the subsystem and the allocator. When the subsystem wishes the allocator to look at the request word, it sets RQ=1. The allocator then comes along, examines the word for requests, attempts to perform the requests, and then sets RQ=0. The subsystem can now examine the word to see what action has been taken by the allocator.

To request storage allocation, the subsystem must set ABC to the desired buffer size (in blocks) and AAD to zero. When storage is allocated, AAD will be set to the FWA/100_8 of the buffer. If the allocator could not allocate the storage at that time (i.e., between when the subsystem set RQ=1 and when the allocator set RQ=0), then AAD will still be zero. The allocator will not allocate storage when AAD is non-zero, or ABC=0.

To release storage, the subsystem must set RBC to the buffer size (in blocks) and RAD to the FWA/100_8 of the buffer. The allocator will set RAD to zero when the storage has been released. The allocator will not release storage when RAD is zero, or RBC=0.

Release and allocation requests can be made simultaneously. A release request will always be honored immediately. No action is performed if AAD≠0 and RAD=0, and/or ABC=RBC=0. 1SD examines the buffer request words and starts the allocator when any RQ=1. If 1SD finds that a subsystem is not loaded (ST=0) and that RQ=0,

ABC≠0, and AAD=0, then 1SD will set RQ=1 and restart the allocator. This procedure is provided to expedite allocation for the subsystems.

6.3 Subsystem Working Storage Area

Six CM words are provided for each subsystem as a working storage area which is always preserved no matter what the subsystem status is. This area is cleared to zero when IOD is initialized.

7.0 LINE TABLE (F)

Each line has a 12 CM word entry in the Line Table which contains all the information pertinent to a given terminal. The lines of a multiplexor will have consecutive 12 word entries. The actual number of entries will depend upon the number of ports of the mux being used and whether or not any ports have a party line hookup. A line table entry has four parts:

* Line Status Word
* Line Request Word
* Line Working Storage Area (from 0-10 words in size)
* Line Message Buffer (from 0-8 words in size)

7.1 Line Status Word (LSW)

12	12	12	12	12
LSB	TSB	RSB	PNB	PLB

This word contains all the status information supplied by IOD for this line. In some instances, the subsystem will contribute status information about the line; however, that is restricted to the first byte (LSB). The subsystem can write the Line Status Word only when TL=1. The detailed breakdown of each 12 bit byte follows:

LSB - Line Status Byte

1	1	1	1	1	1	3	3
TL	IP	EP	BK	LFT	CC	CS	ID

TL : Terminal Lock. When IOD decides the subsystem should do something for this line, IOD sets TL=1 and locks itself out of writing LSW and LRW. The subsystem will be loaded (if ST=0) until TL is reset to zero. When TL=1, IOD is not communicating with the terminal.

IP : In Process Flag. Used by IOD as an internal processing flag. If IP=1 when TL=1, then an error recovery attempt is in process.

EP : Error Processing Flag. Set to 1 by IOD when an error occurs and is equivalent to TL. ISD will log the error, clear EP, and set TL. The subsystems should ignore EP.

BK : For TTYs, set to 1 by IOD whenever a % is entered during output or as first character of an input line. Not used for CRTs.

LFT : First Time Flag. Set to 1 by subsystem zero when a line is assigned to a new subsystem or by the new subsystem when the line is reassigned to subsystem zero. Set to zero by the assigned subsystem (normally after initialization of the line is complete).

CC : Current Cycle Indicator. (Set by IOD).

0 = IOD is currently sending a WRITE or ALERT.

1 = IOD is currently POLLing the terminal.

When TL=1, CC indicates the last action taken by IOD.

0 = WRITE, ALERT, or the subsystem set RD=0.

1 = POLLing, READ received, or errors.

CS : Communications Status. (Set by IOD)

0 = disconnected or not available.

1 = idle (waiting to output SYNC words).

2 = waiting to output (SOM).

3 = outputting (SOM through MPC).

4 = waiting to input (resync).

5 = inputting (SOM through MPC).

6 = input complete (line turnaround and send sync starts).

7 =

ID : Subsystem ID. When ID=0, the line is idle and assigned to subsystem zero. When a new subsystem is requested, subsystem zero will set ID to the number of the subsystem (the relative location of the subsystem entry in the subsystem table) and set LFT=1, thereby assigning the line to the new subsystem. (See Section bb for reassigning lines to subsystem zero.)

TSB - Terminal Status Byte (not altered by the subsystem)

200 USER

1	1	5	1	1	4
WA	OFL		SB		SA

Teletype

4	3
	CP	BY

WA : Write acknowledged indicator.

0 = previous WRITE message has not been acknowledged.

1 = previous WRITE message has been acknowledged.

SB : Sequence bit.

This is current unacknowledged write sequence bit for the 200 USER. When IOD sets WA=1, SB is flipped.

SA : Site address of the 200 USER terminal.

CP : Current relative CM word to WBB $0 \leq CP \leq 7$

BY : Byte position of next output pair of characters plus 3

$3 \leq BY \leq 7$

OFL : Set By IOD/UCP when Terminal off-line

RSB - Reply Status Byte (set by IOD at the end of each cycle)

1	1	1	3	3	3
SR		DR	DS	EC	RC

SR : Current sequence bit received from the 200 USER terminal.

DR : Data Received Flag.

0 = no data was received.

1 = data was received and stored according to the value of RD.

DS : Device Status.

This field contains information pertaining to the E codes received from the 200 USER terminal with the last READ message (RC=1).

0 = none was received. (assume E1 when EC=0)

1 = E1: keyboard SEND key was depressed.

2 = E2: device not attached or not ready.

3 = E3: device ready.

4-7 = reserved and not used.

EC : Error Code.

0 = no errors

1 = too much data received for available buffer space.

2 = bad card code.

} IOD will clear IP when EC ≤ 2

3 = bad header codes.

4 = character parity error detected.

5 = message parity error detected.

6 = lost data

7 = data set timeout: outputting (CS=2 or 3); inputting (CS=4 or 5).

RC : Response code received from the 200 USER terminal.

0 = none was received. Timed out (CS=4) or Unrecognized (CS=5 or 6)

1 = READ: data and/or device status received.

2 = REJ: terminal busy or read request not set.

3 = ERR: parity error detected by the terminal.

4 = ACK: WRITE or ALERT received OK by the terminal.

5 = No PP buffer was available for IOD to store the data.

6 =

7 =

PNB - Port Number Byte (initialized by IOD and never altered)

The byte contains the multiplexor index and port number for this terminal in the following format:

1	3	2	6
TY		MX	PN

TY : Terminal Type.

0 = 200 USER.

1 = Teletype.

MX : Multiplexor index number. $0 \leq MX \leq M-1$ gives the relative position of the mux for this terminal in the equipment table.

PN : The mux port number for this terminal. $0 \leq PN \leq 15$ (6671), or $0 \leq PN \leq 63$ (6676).

PLB - Party Line Byte (not altered by the subsystem)

This byte contains the 12 bit address of the next Line Status Word entry associated with this multiplexor port. This allows several terminals each with unique site addresses (SA) to be connected via a party line hookup to a single multiplexor port. If only a single terminal is connected to this multiplexor port, then PLB will be set to zero by IOD at initialization, and IOD will increment SA, when no response is received from the terminal when ID=0.

7.2 Line Request Word (LRW) (not altered by IOD when SD=0)

12	12	12	12	12
LRB	CBB	JBB	RBB	WBB

This word contains all the information supplied by the subsystem in making a request for IOD action. IOD will not alter any of the information in this CM word when TL=1 or SD=0. Note that no action will be taken by IOD until the TL bit is cleared in the Line Status Word.

LRB - Line Request Byte (set by the subsystem, IOD may clear SD)

2	1	2	1	3	3
SD	CR	RD		MT	DT

SD : Send Data Flag. (used to control all WRITE operations)

0 = no write operation.

1 = send data from jungle unit numbered WBB.

2 = send data from a CM buffer (WBB contains FWA/100_8).

3 = send data from message buffer (WBB contains FWA).

If SD≠0 when TL is set=0, IOD will set SD=0 when the WRITE has been successfully completed. IOD will not alter LRW when SD=0.

CR : Carriage Return Option. (CR is ignored for TTYs)

If CR=1 and SD≠0, a carriage return will be sent to the terminal by IOD prior to sending data.

RD : Receive Data Flag. (used to control all POLL (READ) operations)

0 = do not POLL for data.

1 = place input data into RESPOND's CIRSTAC.

2 = place input data into a CM buffer (RBB contains FWA/100$_8$)

3 = place data into the message buffer (RBB contains FWA).

When RD≠0, IOD will POLL until a READ message is received from the terminal. RD will not be altered by IOD. However, the subsystem may clear RD when SD=0, to stop POLLing. If IOD stops POLLing for this reason, CC will be set to zero.

MT : Message Type. (MT is ignored for TTYs and all outputs are preceded by a CR/LF)

0 =

1 = POLL (SD must =0, IOD will POLL once)

2 = ALERT

3 = WRITE (normal TTY message)

4 = Reset Write

5 = Clear Write

6 = Diagnostic Write

7 =

DT : Device Type. (destination of write message data) 200 USER only

0 =

1 = E1 (display)

2 = E2 (printer)

3 = E3 (card reader)

4 = E4 (display)

5 =

6 =

7 =

CBB - Current Buffer Byte (Altered by IOD)

Used by IOD as an input address or input word count. Ignored by the subsystem except when RD=2.

JBB - Jungle Buffer Byte (Altered by IOD)

Used by IOD as an output address or output word count. Should be ignored by the subsystem.

RBB - Read Buffer Byte (not altered by IOD)

RD=0
or 1: RBB is ignored.

RD=2: RBB contains the FWA/100_8 of the CM input buffer for this line. If RBB=0, and data is received from the terminal, then IOD sets EC=1 and no data is written to CM. RBB must be set by the subsystem, and it will usually be the address AAD obtained from the Buffer Request Word. (See Section 9.0 for a description of the format of the CM buffer.) RD may not be set to 2 for teletype lines.

RD=3: RBB is the absolute address of the input message buffer for this line (relative to RA). When IOD sets LFT=1(the first time flag), then RBB will point to the 7th word in the line table entry, which leaves a Line Working Store Area of 5 CM words. The subsystem may adjust RBB to make the message buffer any size from 0 to 8 CM words. If RBB does not point to within the 10 CM word area immediately following the Line Request Word, and data is received, then IOD sets EC=1 and no data is written to CM. Up to 80 characters may be received.

WBB - Write Buffer Byte (not altered by IOD)

SD=0: WBB is ignored.

SD=1: WBB is the jungle unit number to be used to begin output. Subsequent jungle unit numbers are obtained from RESPOND's jungle unit link table. A zero unit number terminates linking. If WBB=0, a WRITE message will be sent to unlock the keyboard (DT=1) or status the device (DT=2 or 3).

SD=2: WBB is the FWA/100_8 of the CM output buffer for this line. WBB must be set by the subsystem. It will usually be the address AAD obtained from the Buffer Request Word. (See Section 9.0 for a description of the format of the CM buffer.) When WBB=0, no data is taken from CM and a two-character message (plus, blank) is sent to the terminal. If DT=2, this message will not disturb the printer condition; however, it may be

used to determine the printer status. If DT=1, then the "+␣▲" will appear on the screen. SD may not be set to 2 for teletype lines.

SD=3: WBB is the address, relative to RA, of the output message buffer for this line. When IOD sets LFT=1 WBB will point to the 7th word in the line table entry (see also RBB). Although LSW+2≤WBB≤4095, the buffer length cannot exceed 8 CM words. Up to 80 characters may be output. If WBB=0, and the line is a CRT, then the (plus, blank) message is sent. If WBB=0 and the line is a TTY, no data is sent.

When a subsystem wishes to perform a WRITE-POLL sequence, it sets SD≠0, RD≠0 and TL=0. IOD will now attempt the WRITE. If no acknowledge is returned, IOD will POLL and check the sequence bit. If the terminal received the WRITE OK, and only the acknowledge was in error, IOD will set SD=0 and continue to POLL. Otherwise, TL will be set to 1 and the subsystem loaded. When a READ occurs, or when any transmission error occurs, or when the subsystem sets RD=0, transmission will stop, TL will be set to 1, and the subsystem will be loaded. If the subsystem sets TL=0 with SD=0 and RD≠0, only a POLLing will occur. If the subsystem sets TL=0 with SD≠0 and RD=0, only a WRITE will occur. In <u>all</u> cases, when SD=RD=0 after TL=0, TL will be reset to 1 and the subsystem reloaded. Any time TL=1 and the subsystem is not loaded, it will be reloaded. If the subsystem desires to have IOD continue action after an error stop, it need only set TL=0.

7.3 Line Storage Area

From 0 to 10 CM words are provided immediately after the Line Request Word as a subsystem working storage area for this line. This area is cleared to zero when IOD is initialized. The actual area is defined to be that portion of the 10 CM words after the Line Request Word which are not used for the input or output message buffers. When a subsystem is initially assigned (i.e., IOD sets LFT=1)the line storage area has five words, the first five words in the 10 CM word area.

7.4 Line Message Area

Whenever SD or RD=3, RBB and/or WBB define the location of the message areas. The input message buffer beginning at RBB extends to RBB+7 or the end of the line table entry area, whichever comes first. The output message buffer beginning at WBB extends to WBB+7 or the end of the line table entry area, whichever comes first. However, the first end of data code in the buffer will terminate before 8 CM words. If neither SD or RD=3, then a line message area does not exist.

RBB and WBB must not equal LSW+12. Note, however, that WBB may be greater than LSW+12, but less than 4096. This allows common messages to be placed in CM above the CM storage area for IOD, which then can be output to all terminals. A 0000 byte will cause a CR/LF to be output for DT=1,3, or 4, or an EOL for DT=2. Received CR/LF codes will be ignored.

No Data on This Page

9.0 CM BUFFERS (H)

CM buffers begin on a 100_8 CM word boundary. The buffers are treated in a linear fashion starting from the first word (word zero).

For output, IOD will transmit the data starting from word zero up until an EOD (end-of-data) code (a 0001 byte) is encountered within the data. The transmitted data will be interpreted in 12 bit bytes as follows:

Byte Value$_8$	Interpretation
0000	DT = 1,3,4 send a CR/LF code. DT = 2 send an EOL (end-of-line) code.
0001	EOD, stop transmitting.
0002	Should not be used.
0003	Send a compression code for 4 zeros.
0004	Send a compression code for 6 zeros.
0005	Send a compression code for 8 zeros.
0006	Send a compression code for 10 zeros.
0007	Send a compression code for 12 zeros.
0010	Send a compression code for 14 zeros.
0011	Send a compression code for 4 blanks.
0012	Send a compression code for 6 blanks.
0013	Send a compression code for 8 blanks.
0014	Send a compression code for 10 blanks.
0015	Send a compression code for 12 blanks.

Byte Value$_8$	Interpretation
0016	Send a compression code for 14 blanks.
0017	Send a compression code for 16 blanks.
0020	Send a compression code for 18 blanks.
0021	Send a compression code for 20 blanks.
0022	Send a compression code for 22 blanks.
0023	Send a compression code for 24 blanks.
0024	Send a compression code for 26 blanks.
0025	Send a compression code for 28 blanks.
0026	Send a compression code for 30 blanks.

For bytes of the form 00YY with $77 \geq YY \geq 27$, a blank and then the display coded character YY are sent. For bytes of the form XXYY with $7777 \geq XXYY \geq 0100$, the display coded characters XX and YY are sent with YY=00 being sent as a blank.

A compression byte (0003-0026) should not be the first byte of data or the first byte following an EOL (0000).

No printer compression will be performed by IOD. It is the responsibility of the subsystem to examine the output data, insert the compression codes, and determine how much data will be output. IOD will continue to transmit until the EOD is encountered regardless of whether or not the screen size is exceeded.

For input, IOD will store the received data starting from word zero. When the read is complete, and if data is received (DR=1), CBB will contain a count of the number of words stored in the buffer. This count will never be greater than 104_{10}. (This necessitates having a 2 block or 128_{10} word buffer before requesting a read). The end of the input data will be indicated by an EOD byte (0001) or by the word count, whichever is encountered first. CR/LF codes that are received will be ignored.

Card input data (DT=3) will be stored in consecutive 8 cm word segments without a trailing 0000 byte or zero word. No trailing blank deletion will be performed. EOR and EOF cards will be indicated by the first byte of an 8 cm word segment as follows:

First byte	Interpretation
000X	EOR card
004X	EOF card

X = The high order digit of the record level number (0 or 1). The second digit appears in display code in the high order character position of the next byte.

No Data on This Page

No Data on This Page

No Data on This Page

10.0 Terminal Idling

A terminal (i.e., an entry in the line table) is considered to be idle when it is assigned to subsystem zero. That is, when the ID field of the line status byte (LSB) of the line status word (LSW) is set to zero. It is always the responsibility of the subsystem currently assigned to a terminal to reset the line status and line request words for subsystem zero. The procedure is as follows:

1. If the terminal lock (TL in LSB) is set, go to Step 3, else go to Step 2.
2. If the send data flag (SD in LRB) is non-zero (i.e., output is in progress), go back to Step 1, else set the read data flag (RD in LRB) to zero and go back to Step 1.
3. Reset the line request word (LRW) as follows:
   a. If no particular action is required, set WBB=RBB=0. Subsystem zero will issue the terminal idle message (with a clear write on CRT's) after reassignment is complete.
   b. If it is desired to issue a final message, set LRB=6031_8 (or 7031_8 if an initial carriage return is desired), RBB=0, WBB=address of the final message. Subsystem zero will issue the message and then issue the terminal idle message (with a normal write on CRT's) after reassignment is complete.

c. If a new subsystem name is already available, set LRB=0031_8, RBB=address of the new name, and WBB=0. Subsystem zero will check the new name after reassignment is complete.

d. If a new subsystem name is already available and it is desired to issue a final message from the message buffer, set LRB=6031_8 (or 7031_8 if an initial carriage return is desired), RBB=address of the new name, and WBB=address of the final message. Subsystem zero will issue the message (with a normal write on CRT's) and then check the new name after reassignment is complete.

e. If a new subsystem name is to be input (or repolled), set LRB=0631_8, RBB=address of the buffer to receive the new name, and WBB=0. Subsystem zero will poll for the new name and check it after reassignment is complete.

f. If it is desired to issue a final message and then wait for a new subsystem name to be input, set LRB=6631_8 (or 7631_8 if an initial carriage return is desired), RBB= address of the buffer to receive the new name, and WBB= address of the final message. Subsystem zero will issue the message (with a normal write on CRT's), poll for the new name, and check it after reassignment is complete.

4. Finally, set LSB=4200. This begins the reassignment to subsystem zero.

Note: In Step 3 above, RBB and WBB must lie in the range of LSW+7 to LSW+11.

Reassignment to subsystem zero will now occur in two steps. With TL=1, LFT=1, and ID=0 (i.e., LSB=4200) the line is locked onto 1SD. 1SD will check the error code (EC in RSB). If EC=7, indicating the terminal has been disconnected, 1SD will reset LRW and repeatedly request 1OD to check the status of the terminal. However, if EC is not initially set to 7, 1SD will clear LSW+2 through LSW+6 and then clear LFT, thereby locking the terminal onto 1ID, the PP program for subsystem zero. In the case where the terminal was disconnected (EC=7 initially), but finally becomes reconnected, 1SD will clear LFT but set LRW so that option 3.a. above will be executed by 1ID.

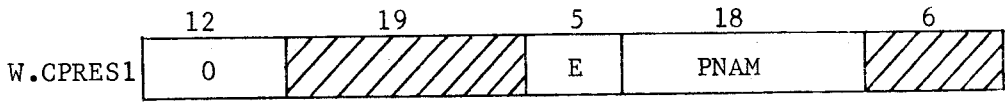
When 1ID is locked onto the line, it will clear TL and allow 1OD to execute the final request from the previous subsystem, and then issue the terminal idle message (if RBB=0) or check the new subsystem name (if RBB≠0). From this point on, 1ID enters its normal sequencing of sending out the terminal idle message, waiting for an input, and sending out subsystem not available if the input name is not identifiable. When an identifiable name is input, 1ID will set the first time flag and the correct subsystem ID for the requested subsystem, thereby assigning the line to the new subsystem. The input line, of which the first 4 characters were identified, is stored in LSW+7 to LSW+11 for further checking by the new subsystem. The one exception to this is RESPOND. For RESPOND, the input line is repolled into CIRSTAC, as well as

being left in the line table area. (For TTY's, 1OD performs a special pseudo repoll, taking the data in LSW+7 to LSW+11 and transferring it to CIRSTAC. LRB=6210_8 for this operation.)

If at any time while a line is assigned to subsystem zero the terminal should disconnect (EC=7), 1ID will set the first time flag, thereby transferring the line back to 1SD. 1SD will then cause 1OD to periodically check for new activity, as described previously, and ultimately clear the first time flag when the line is reconnected.

11.0 Control Point Drop

Dropping of the control point may be requested either by the central site operator entering an n.DROP, or by any program at the control point detecting some form of fatal error. In either case, the final dropping of the dedicated PP containing 1OD is controlled by the word W.CPRES1, in the CM control point area, and the control point error flag.



W.CPRES1 is initially set to zero by IOD. Periodically, 1OD examines the high order 12 bits of PNAM. If these are still zero, 1OD continues processing. Otherwise, 1OD stops all communications, requests 1PR to be loaded, and then waits for 1PR to transfer a new copy of PP resident over the channel previously used for I/O. During this waiting period, 1OD issues monitor pause requests to permit storage moves.

After being loaded, 1PR obtains the channel number for the transfer from its input register and stores the number in the required channel instructions. Next, 1PR examines the high order 6 bits of the PNAM field in W.CPRES1. If these are non-zero, PNAM and E+1 are stored in the SYSTEM FAILURE message, which is then issued to the control point dayfile. At this point, regardless of the state of PNAM, 1PR waits until the operator drop flag is

set before proceeding further. Calls are made to R.PAUSE during the wait to permit storage moves. When the drop flag is set, 1PR waits until the channel is activated, disconnects it, and then waits until it is activated again before outputting the resident to 1OD. 1OD, conversely, initially activates the channel, waits for disconnect, and then re-activates the channel to provide synchronization between the two PPs before the transfer takes place. When the channel becomes empty (transfer complete), 1PR drops its PP, while 1OD first drops the channel and then drops its PP.

1SD, the transient PP program in monitor recall used to load and service subsystems, also monitors the high order 12 bits of PNAM and will drop immediately, without requesting monitor recall, when these become non-zero.

11.1 Fatal Error Drop

When a PP program associated with the system detects an irrecoverable error, such as an R.TFL failure, the following procedure will be followed:

1. W.CPRES1 will be set so that PNAM contains the name of the PP program and E contains some value indicating the type of error.
2. A wait loop will be entered which periodically calls R.PAUSE and then checks for operator drop.
3. When operator drop is set, the PP will be dropped.

When the above procedure is executed, three PPs will become "hung" at the control point (1OD, 1PR, and the error PP), and the message

SYSTEM FAILURE XXX Y

will appear in the dayfile and at the control point. XXX will be the three character name stored in PNAM and Y will be an alphanumeric character corresponding to the value of E+1 (i.e., E=0 produces A, E=1 produces B, etc.). The central site operator may now drop the control point, causing all three PPs to drop, or may wait until all the other control points are clear and then take a deadstart dump of the PPs that were hung.

Any PP program which is in monitor recall in place of, or in addition to, recall via 1SD, must also periodically examine W.CPRES1. When the high order 12 bits of PNAM become non-zero, the program must drop immediately without requesting further recall. Programs

which are recalled only via 1SD do not have to examine W.CPRES1 as this function is performed by 1SD.

11.2 Normal Operator Drop

On a normal operator drop (i.e., no fatal errors have been detected), subsystem zero (1ID) ultimately sets W.CPRES1 after all lines have been reassigned back to subsystem zero. As each line is reassigned, 1ID issues an operator drop message to the terminal and then "logically" drops the line. When all lines have been dropped, 1ID sets the high order 12 bits of PNAM in W.CPRES1 to 0077_8, and then drops. Since the operator drop flag is set and the high order 6 bits of PNAM are zero, the transfer of the resident from 1PR to 1OD occurs immediately and no error message is issued.

When a normal operator drop is requested, 1SD and 1ID perform slightly differently than as described in Section 10.0 . 1SD ignores checking the error flag (EC in RSB) and immediately clears the first time flag (LFT) so as to lock the terminal onto 1ID. 1ID will ignore retransmissions on errors, and will immediately set the terminal to be "logically" dropped if error code is set to 7 (line hung up or turned off). Requests to issue a final message will be performed, but any name-checking will be ignored and the operator drop message issued.

Part II - INTERNAL SPECIFICATION

1.0 I/O DRIVER PHILOSOPHY

The voice grade I/O Driver (IOD) supporting RESPOND II, SENTRY and EXPORT/IMPORT 200 is a dedicated PP program which, after loading and initialization, remains in its PP and active until the control point word W.CPRES1 is non-zero. IOD does not use the normal PP resident functions once it is in normal operation. Because of the time critical nature of the service requirements of a clocked data set (201), IOD cannot depend upon the sometimes slow response of MTR. As a result, only one MTR function is used by IOD: "pause for relocation". This is necessary to keep the rest of the system operational. IOD will check the storage move flag at least every 3.3 milliseconds and place a "pause" in its output register as appropriate. At this time, IOD will also set a "move in process" flag in a direct cell (MOVE) in its PP. IOD will continue its normal operation, except that no reads and writes to CM will take place. If any data input operations are taking place at the time of a storage move, IOD will attempt to buffer in the PP as much data as possible until the storage move clears. If this is unsuccessful in the case of a 200 USER, IOD will POLL the terminal again for a re-transmission of the READ. IOD will be designed so that no TTY data will be lost for "normal" storage move situations (up to about 300-400 milliseconds, worst case).

IOD is always active. It services both 6671 and 6676 multiplexors. Each of the ports of the multiplexors is operated independently. That is, line 0 may be outputting while line 1 may be inputting. Also, a line may not be communicating, while IOD awaits some action from the subsystem driving the line. At IOD initialization time, each line or terminal is considered to be unassigned. The initial operation of the driver on each line is to assign the line to some subsystem (e.g., EXPORT/IMPORT 200). The driver sends a message to the display of the 200 USER indicating "TERMINAL IDLE". IOD then POLLs the line indefinitely until the user enters a keyboard request (e.g., IMPORT) and presses the SEND button. When the request is received, a transient PP program 1ID (subsystem 0) compares it with a list of 4-character call names which are stored in CM. Each of these names corresponds to a subsystem. If a match is found, the id number of the subsystem ($1 \leq id \leq 7$) is inserted into a table entry for this line (Line Status Word). If no match is found, the message "SUBSYSTEM NOT AVAILABLE" is displayed.

The table in CM which contains the Line Status Words (LSW) has an entry for each terminal of the system. The system allows a party line hookup such that more than one terminal may be physically connected to a multiplexor port at one time. In this case, several entries of the LSW may all be associated with one multiplexor port. IOD will service these terminals in a round-robin

fashion, such that each gets nearly equal service.

The normal sequence of operations is such that the subsystem sends a message to the terminal, IOD waits until a reply is received, then IOD stores the received message in CM, sets a "terminal lock" flag, and the PP program of the subsystem is loaded. Optionally, the subsystem can consider itself always loaded ST=1, in which case a PP program is not loaded each time (e.g., RESPOND II). This normal sequence is started when IID assigns a subsystem to the line and gives to the subsystem the initial call name message. In practice this message may contain additional information for the subsystem (e.g., passwords).

1.1 Data Handling

The message exchanged with the 200 USER Terminal consist of two parts. The first part comprises the SYNC, SOM, site address, station address, control code, E codes, ETX and MPC, which are essential, but overhead characters. The second part is the data, which may be included with either a WRITE or READ message. Except for the 8 and 12 bit diagnostic modes, all of the items of the first part are generated automatically by IOD. A subsystem need only set a few bits in the Line Request Word and the I/O Driver will exchange messages with the terminal and process all of the first part internally. If data is to be sent to the terminal, the subsystem stores the data as 6 bit Display code in CM, sets up the Line Request Word and clears Terminal Lock (TL) in the Line Status Word. The driver will format the proper message including the data and transmit it to the terminal. In response, the terminal may send back to IOD a READ message which contains data from the display (keyboard) or the card reader. IOD strips off the first part and converts the data part to 6000 display code and then stores the converted data in CM for the subsystem.

In summary, the subsystem is only concerned with handling display coded data and requests IOD to send or receive data via bits in the Line Request Word and Line Status Word. All errors which

require re-transmission (e.g., parity, no response, etc.) are processed automatically by the driver when requested by the subsystem.

Considering the large number of terminals which the driver can handle simultaneously, the most critical resource of the driver is PP memory space. This is caused in part by the timing requirements of the clocked (201) data sets. 201B data sets require servicing every 3.3 milliseconds, which does not allow enough time in the PP to read CM to get every data character when it must be transmitted. In fact, with 32 terminals operating there is not time to fetch data in chunks as small as 1 CM word. The best compromise seems to be an 8 CM word block which conveniently is the size of RESPOND's jungle unit buffers.

IOD has a pool of buffers in the PP memory. In PP words the buffers are 40 words long. IOD is designed to operate with fewer than 32 buffers in the pool. This, of course, would mean that 32 terminals could not be inputting and/or outputting data simultaneously (i.e., instantaneously). In fact, as the number of terminals actively transmitting or receiving data approaches the number of buffers, the performance of each terminal will begin to degrade. Studies show that this degradation will become apparent to the EXPORT/IMPORT user immediately, while the RESPOND

user may not notice it at all. The reason is as follows: When EXPORT/IMPORT has control of the terminal, the speed of the terminal is limited by the speed of the transmission line. Therefore, a high percentage of the time, data is being transmitted. When there is a lack or shortage of buffers for IOD, it must temporarily suspend data transmission. The printing rate will therefore suffer at the terminal. 200 USER Terminal equipped with the 100 CPM card reader will not notice the reduction in card reading rate as soon, since reading is card reader limited and not line limited. The terminal in the RESPOND mode is characterized by a rather low utilization. Studies show that the think-time, or head-scratching time, of the user is in the tens or twenties of seconds. In addition, the keyboard entry time of the user is buffered in the 200 USER display and data transmission does not take place until the user hits the SEND key. Typical messages are short (e.g., 20 or 30 characters), so that a data buffer of the PP is needed for only 50 to 100 or more milliseconds during a 20 second period, for example. Therefore, on the average, only a very few number of buffers are needed to provide excellent response to the RESPOND user. It is expected, therefore, that the response time of the RESPOND user is limited by the computing capabilities of the RESPOND central program rather than by buffer limitations of the driver. There is an exception to this, however, if the number of EXPORT users exceeds the number of buffers.

1.2 Buffer Handling

As was mentioned before, data can only be transmitted to or received from a terminal if a PP buffer is available for the storage of the data. A scheme was developed so that buffers can be obtained from, and returned to, the pool of available buffers in a very few number of microseconds. Therefore, the acquisition of a data buffer can be accomplished on the fly between the transmission of characters to the multiplexor. At this point, a nice feature of the 200 USER Terminal should be mentioned, which makes some of the driver design possible. The 200 USER will ignore (i.e., throw away) any SYNC codes which appear in the data message from the central computer. This makes it possible for the driver to occasionally "pad out" the data message with SYNC codes when it reaches some kind of a temporary impasse and cannot get data from CM (e.g., storage move or no buffer available).

In the output (to the terminal) phase of communications, a buffer is obtained for a line at the last possible moment. When the line is being turned around and the SYNC codes, SOM, etc., are being sent out, there are no buffers assigned to the line. However, when the WRITE control code is being sent (it is the character immediately preceding the data), a request is made of a routine named READCM to obtain a buffer for the line and read a block of 8 CM words from CM. Once during each multiplexor cycle (which must be less than 3.3 or 4.0 milliseconds to keep the data sets

happy), READCM is entered to fetch one buffer and read 8 CM words. If no buffers are available, READCM just exits. During the next cycle, if no buffer was obtained, a SYNC word is sent instead of the data character. However, if a buffer was available and there was no more than one request made that cycle, and a storage move was not in process, then the read is completed and the data is available in a PP buffer for transmission to the terminal. This should keep that terminal happy for at least 80 cycles, or until the end of data, if less than 80 characters are to be sent (8 CM words = 40 PP words = 80 characters). If two or more terminals required buffers for output during the same cycle, the higher numbered terminal will get the buffer and the other terminals will be sent a SYNC word instead. This operation will tend to stagger the data transmission so that later requests for buffers in a long transmission will occur in different cycles. As soon as the 80th character is sent, the buffer is immediately returned to the pool, and at the same time a request is made to READCM for another buffer and a CM read. If the buffer contained fewer than 80 characters, the buffer would be returned after the last character was sent. In the case of short messages to the display (e.g., typical RESPOND output), the buffer will be held by the line for only a short while.

In summary, data output to a 200 USER can continue even with buffer shortages, and frequent storage moves by outputting SYNC words during the unavailability of data. Because the same is not true of data input, only one buffer is used each cycle by READCM, and then only if no writes to CM are required. During periods of high buffer activity, the chances are improved that a buffer will be available for input data.

Because of the built-in output priority of high numbered lines, and the short nature of RESPOND output, and the long nature of EXPORT output, predominantly RESPOND terminals should be on higher numbered lines than EXPORT terminals. The following example will illustrate:

Given 32 terminals and 16 (for illustrative purposes) buffers in the PP, and the highest number 16 lines are all printing in the EXPORT mode; when the terminals are in phase, all 16 buffers would be monopolized by the EXPORT terminals, and any RESPOND output message would have to wait for up to 4 seconds for the close to 1000 character outputs to be completed on at least one line. However, with RESPOND users on high numbered lines, a buffer would be available as soon as any EXPORT terminal had completed an 80 character block. In this case, the wait would be 300 or 400 milliseconds, and the EXPORT terminal would be sent SYNC words for the duration of the RESPOND data message.

Note that a low numbered EXPORT terminal will experience serious degradation of printing when more terminals are printing than buffers exist.

In the input (from the terminal) phase of communications, again a buffer is obtained for a line at the last possible moment. After the terminal has been POLLed, the driver waits for the READ message from the terminal. The data characters follow the READ control code received from the terminal. Sometimes READ messages are received from the terminal which do not contain data. One case might be when the user hits the SEND key to signal the subsystem to take some action (e.g., display another PAGE under RESPOND). The READ message contains only the E1 code indicating a keyboard message. The driver does not require a buffer in this case. A more typical situation occurs during printing. After each large block of printer data is sent to the terminal, it is necessary for the driver to POLL the terminal for the status of the printer. This comes back in the form of a READ message with either an E2 (not ready) or an E3 (printer ready) code. Since no data is included, no buffer is needed.

Typically, for RESPOND when IOD POLLs the terminal, a REJ (reject) is received because the user has not hit the SEND key. However, when he does hit the SEND key, the READ message will be sent and

it may contain data. When IOD determines that data is arriving after the READ control code is received, a buffer is immediately obtained for the line from the pool. If none is available, the entire transmission is lost and the driver will automatically receive the data, throw it away, and then rePOLL the line as soon as possible. Note that the unavailability of a buffer could occur in the middle of a message as well as at the beginning.

As soon as the buffer is filled (i.e., 80 characters), the buffer is placed in a queue for subsequent writing to CM. The design of the driver assumes that it may not be possible to write that buffer to CM in time to re-use the same buffer for the later portions of the READ message. It is easily possible that several lines are inputting in phase and the queue for WRITCM (the routine that writes to CM and releases buffers to the pool) would be too long to completely process during the time available in one cycle. WRITCM does, however, have first priority over READCM in that READCM will not be called if the WRITCM queue is not zero length. The nature of the data sets and card reader performance is such that IOD cannot predict exactly when a READ message will come back in RESPONSE to a POLL. Therefore, IOD cannot successfully stagger READs so that the queue to WRITCM would always be nice and short. On the average though, since the number of terminals actively transmitting data is low with respect to the

number of characters in a buffer (i.e., 80), the WRITCM queue should be short. The result should be that each call to WRITCM each cycle will generally empty the queue and make the previously filled buffers available for the next character to be input. If all the writes to CM cannot be completed, then things will still continue OK as long as the pool of buffers does not dry up.

During a storage move, WRITCM will not be able to operate, and buffers will gradually be used up. However, on the average, some lines may be outputting, and periodically they will release empty buffers to the pool to help sustain the input operations on other lines. (Note that READCM will not operate during a storage move.)

A critical situation occurs when most of the lines are beginning an input at the time of a storage move. If the move takes too long and all buffers are used up, then data may be lost on all lines, and the driver will have to rePOLL the lines to obtain the data.

If the driver is attached to a control point such as 1, which only moves during an EDITLIB, and they occur rather infrequently, then the re-transmissions required should be a low percentage of the total number of transmissions.

2.0 I/O DRIVER INITIALIZATION

The initialization of the I/O Driver is done in several phases in order to minimize the amount of storage used by one-time-only code. The first phase, PP Program IOD, does a number of miscellaneous tasks before loading the main overlay 1OD on top of itself. The second phase does the actual initialization of code in 1OD and terminates with the loading of overlay 5OD, the conversion tables. The third phase loads 1SD into a transient PP in order for it to go into MTR recall.

Phase 1

When IOD is loaded, its PP input register must contain the location SYSPTR-1 in byte 4 and AR in byte 2 (see Section 3.0 of Driver IRS), where bit 2^{7-SS} of AR is 1 when subsystem SS is available.

IOD requests the assignment of the appropriate equipment (6671 and/or 6676) and channel(s) from MTR and verifies that the equipment is actually functioning.

Phase 1 {Continued}

The equipment and line tables in central memory are set up from information supplied in the multiplexor port definitions and the EST entries. The subsystem table is set up from the SYSNAME table and AR. The actual S and T values are written to SYSPTR.

CS is set to zero for CRT lines; it is set to 4000B for TTY lines on a 6671 and to 4100B for TTY lines on a 6676. WAIT is set to MTAR and OUT to BLAST for all lines attached to a 6671; OUT is set to 6000B for 6676 lines.

On exit {i.e., when 10D is loaded}, direct cell DFM contains the location of the second line of the dayfile message area; CMP contains SYSPTR-1, FLAG, JWRITE, JNEXTW, MOVE, and JUNGCT are zero; JNEXT is = 1; BLANKS is = 5555B; and LL contains the actual number of ports being used.

Phase 2

This phase of the initialization is assembled in the area which will eventually be occupied by the conversion tables; on completion of this phase, overlay 50D is loaded

Phase 2 {Continued}

into the area where code had just been executed.

Direct cell JEMPTY is set to JUNGL, JNEXT to JUNGL+1, and JUIN and JUOUT to JURD.

ITASK is set to TLOOPE for TTY lines and to NEXT for CRT lines. CS is set to 40008 for all TTY lines. OTASK is set to INTX for all lines attached to a 6671; it is set to SCAN for lines attached to a 6676.

The 6676 and 6671 channel numbers are inserted into the I/O instructions and the appropriate OUT and IN addresses are inserted into the OAM and IAM instructions. One of the multiplexor channel numbers is inserted into the channel instructions in the DROP routine.

The number of ports -1 in all preceeding multiplexors is stored in LT for 6676's. LPP contains the total number of ports on all multiplexors. LPP is stored in L2 for WRCM.

Phase 2 (Continued)

The control point number is inserted into the 1PR and 1SD call words; also, SYSPTR-1 is inserted into the 1SD call word.

JUNGLX is set to JUNGLI.

The JULAE, CIRSTACK and JUNGLE pointers are inserted into JULP, CIRP, and JUNP, respectively.

Overlay 50D is loaded.

Phase 3

M.RPJ is stored in location MXRPJ.

An eight-PP-word buffer is assigned for each TTY in the system. The location of this buffer (or zero in the case of a CRT line) is stored in BUFIN.

1SD is loaded into a transient PP in order to put up the request for MTR recall.

The phase clock PCK is initialized from the system clock and processing begins.

3.0 MAIN PROCESSING LOOP NARRATIVE

Once the driver has been initialized, execution proceeds through the main processing loop which performs the four main activities of the driver.

1) I/O to one or more multiplexors.
2) Sequencing and execution of the input and/or output tasks for each line.
3) Maintaining the SCOPE interface, and,
4) Moving data to and from Central Memory.

Because of the timing requirements of the clocked 201 Data Sets, this loop must be completed at least once every 3.3 or 4.0 milliseconds. If at least one 201B Data Set is used, then 3.3 milliseconds is the required time. Even if 201 data sets are not used, the loop should be completed at a fast enough rate to process TTY data to and from CM. The system is designed so that the most probable worst case timing for 32-200 USER terminal ports or 128 TTY terminal ports would not exceed 3.333 milliseconds. See the Timing Considerations section for a more detailed description.

The following description of the main loop is broken up into convenient pieces which are titled with labels from the code.

FLOOP: {6676 I/O}

This section is not assembled if M76 = 0. When M76 = 1 or 2, then 6676 multiplexors may be used and this section performs the status and I/O operations for the 6676's.

MM76 contains the number of 6676 multiplexors which were obtained from the EST for this run. If MM76 = 0, then control goes directly to MUX10. Otherwise, MM76 is stored into MM as the MUX index and the first 6676 is functioned for select status request. If the channel is still active after the function, then the 6676 did not respond, a dayfile message ▽NO MUX▽ is written to the operator, the channel is disconnected and the status is retried. If a solid MUX or channel failure has occurred, then the driver will hang in this loop.

If the 6676 does disconnect the channel then the driver activates the channel and inputs the 12-bit status word, and stores it in the MUX status table. The 2^1 bit (input required) is tested and if not set, indicating no input is required, then con trol goes to F622 where the next 6676 is processed. However, when "input required" is set, the 6676 is functioned for select output and the appropriate words from the table OUT are written to the MUX. Then the 6676 is functioned for select input and the correct number of words are accepted from the MUX and stored into the appropriate locations in the table IN. The driver initialization sets up the correct counts and locations directly into the inline code. Finally the current 6676 port number in the table LX is reset to the highest number port on the MUX from the table LT. Note that if all the TTY tasks are not completed within 100 milliseconds for this 6676, that input data may be lost or output data may be repeated for the lowest numbered ports.

At F622, MM is decremented, and if zero, control goes to MUX10. If however, two 6676's are being used, then the second 6676 is processed in an identical manner beginning with the function for select status request, as described above.

MUX10: {6671 I/O}

This section is not assembled if M71=0. When M71=1,2,3, or 4, then 6671 multiplexors may be used and this section performs the status and I/O operations for the 6671's.

MM71 contains the number of 6671 multiplexors which were obtained from the EST for this run. If MM71=0, then control goes directly to ENDMUX. Otherwise, MM71 is stored into MM as the 6671 MUX count and the first 6671 is functioned for select status request. If the channel is still active after the function, see the description under FLOOP for the 6676. Otherwise, the channel is activated by the driver and the 12-bit status word is input and stored. If the 2^0, 2^4, or 2^5 bits are set {service failure, output failure, or memory parity error} then a °MUX=XX° error message is written to the operator. In any case, the 6671 is functioned for select output and the appropriate words from the table OUT are written to the MUX. Then, the 6671 is functioned for select input and the correct number of words are accepted from the MUX and stored into the appropriate locations in the table IN.

MM is decremented, and if zero, control goes to ENDMUX. Otherwise, all remaining 6671's are processed in an identical manner beginning with the function for select status request as described above.

ENDMUX: {Switch to 6671 task processing}

At ENDMUX, the channel 14 clock is read. The number of whole milliseconds since the last loop is stored in the d portion of the instruction at MSEC. The 24-bit millisecond clock {CK} is updated and if the MOVE flag is zero, the multiplexor status table is written to CM.

If M71 or MM71=0, then no 6671 are being used and control goes to NEXT10. Otherwise, the 6671 task sequencing loop is initialized. The current line number L is reset with the number of 6671 ports from EL20. And the multiplexor mode flag is set = 7000_8, indicating the 6671.

L00P: {6671 task sequencing}

At L00P, the appropriate input or output task is entered for each 6671 terminal. Normally, the total time spent at L00P will be less than 1.6 milliseconds, which would allow data to be moved to or from CM during the same main 3.3 millisecond loop. However, under worst case task timing, the time at L00P may reach 2.8 milliseconds.

The high order bit of the entry in the table CS is tested and if set control goes to TL00PA to process the TTY line. If not set, the line is a 200 USER and if CS=0, control goes directly to NT where the next 0TASK is selected and entered. 0therwise, ▽data rejected▽ bit is tested from the MUX input word in table IN and if set, control goes to REJECT. At REJECT, the millisecond count between characters in the table WAIT is reduced by MSEC and if the result is still positive, control goes to NEXT. Else, a time out error has occurred {too much time elapsed between characters on a clocked line} and control goes to TIMERR.

If the ▽data rejected▽ bit is not set, then the high order bit ▽valid data▽ is tested. If neither are set control goes to NT, where the next output task address

is fetched from 0TASK and control goes to it. If the ▽valid data▽ bit was set, then control goes to VAL. At VAL, the 8 low order bits from IN are the data item and are saved at TEMP. A check is made for ETX and if so, control goes to PETX. Else, WAIT is reset to CHWAIT and the next input task address is fetched from ITASK and control goes to it.

All output tasks return to NEXT1 after their completion. At NEXT1, WAIT is reset to CHWAIT, and control goes to NEXT.

NEXT is the end of the 6671 task sequencing loop. L is decremented by one to advance to the next line. If L is still greater than or equal to zero, control returns to L00P, where the next line is processed. 0therwise, control goes to NEXT10.

NEXT10: {Switch to 6676 task processing}

The multiplexor mode flag is set = 0, indicating the 6676 and control goes to TL00P.

TL00P: {6676 Task Sequencing}

This section is not assembled if TTY=0. If MM76=0, then no 6676 multiplexors are being used this run and control goes to EXTRA. Otherwise, for MM76=1 or 2, MM76 is stored into LM76 as the multiplexor index. Each time that main control comes through TL00P only one TTY on each 6676 is processed.

At TL00PZ, the current line number for this 6676 is obtained from the table LX and stored in L. The value is compared with the value in table LT to determine if all TTY have been processed for this 6676. If L equals LT, then control goes to TNEXT. {See Below}.

TL00PA is the beginning of the task processing for a given TTY line. The "valid character" status bit is checked in IN and if set, the 8 data bits are extracted from IN and stored in TEMP. The current input processing task is fetched from ITASK and control goes to it.

If "valid character" is not set, then the "data rejected" status bit is checked. If not set, then OUT is cleared and the current output task is fetched from OTASK and control goes to it. Otherwise, when the "data rejected" bit is set, control goes to TL00PH, where the multiplexor mode flag is tested. If the mode is 6676 and a disconnect function is in the OUT table, then control goes to the current OTASK. Otherwise, control goes to TL00PE, which is where all OTASK's return control. {Input processing tasks always

return control to TL00PE

At TL00PE, control returns to NEXT for the 6671 mode. Otherwise {6676 mode}, the current 6676 TTY line number in the LX table is decremented by one. Then at TNEXT, LM76 is decremented by one. When LM76=0, there are no more 6676's to process and control goes to EXTRA, else control goes back up to TL00PZ where the next 6676 is checked.

EXTRA: {SCOPE Interface}

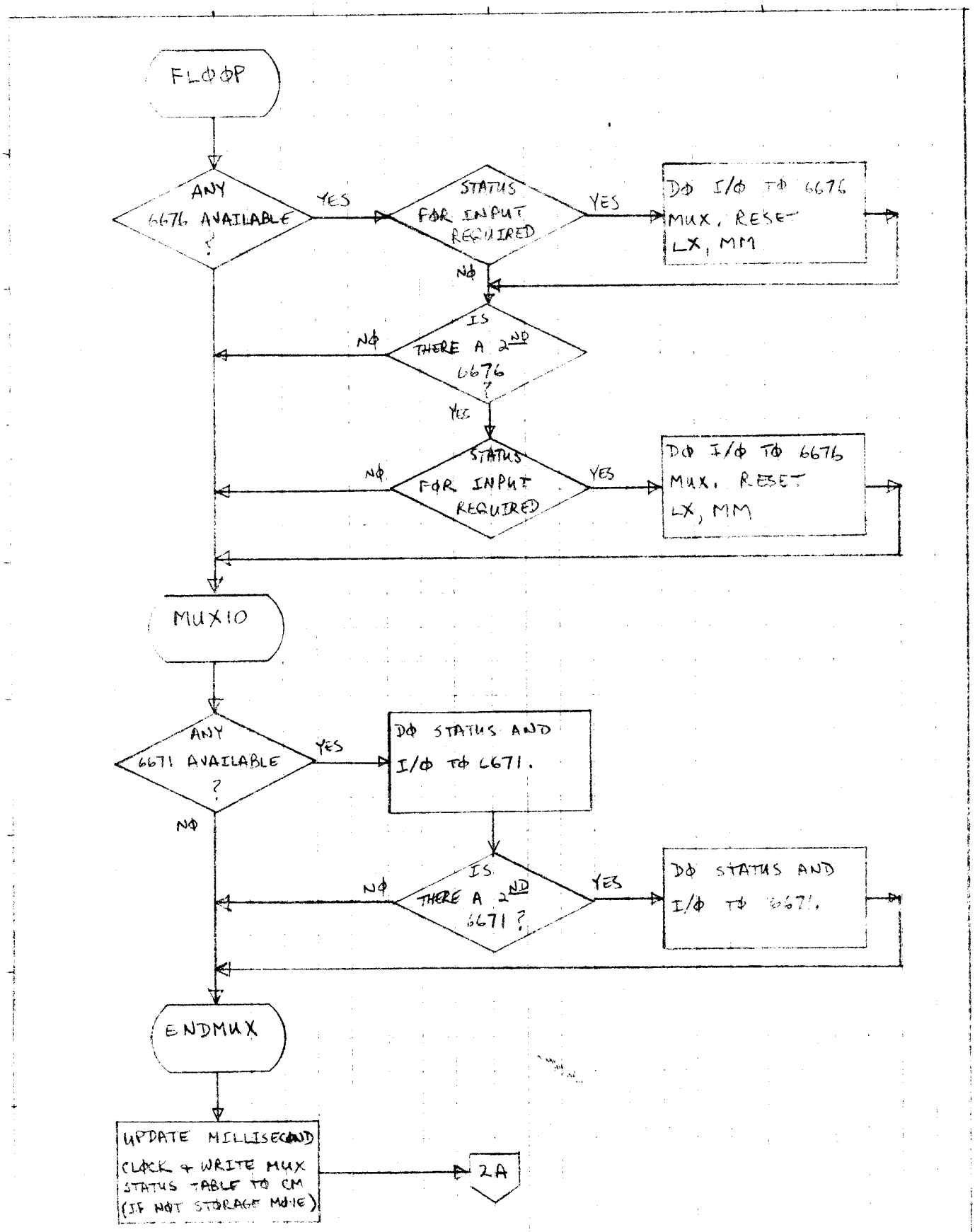
The output register of the PP is fetched from CM and tested for zero. If non-zero, then a monitor pause must be there and control goes to FL00P. After initialization, the only request ever written to the driver's output register is a pause {M.PAUSE}. Otherwise the control point status word is read from the control area and the current RA and FL are saved in the direct cells D.RA and D.FL. The storage move flag is placed in direct cell MOVE. If MOVE is non-zero, then M.PAUSE is written to the PP's output register and control goes to FL00P. Otherwise the Respond word {W.CPRESI} is fetched from the control point area and the system drop flag is checked in the low order byte. If non-zero, then control goes to DROP. At EXT2, the clock is checked to see if enough time remains during the current cycle to attempt a call to the system routines WRITCM or READCM. If enough millisecond remain then WRITCM is entered. If WRITCM processed data, it returns with the A-register non-zero and

control goes to FLOOP. Else JUNGCT is checked and READCM is entered if JUNGCT=0. If READCM was called then control goes to FLOOP. When neither READCM or WRITCM is called, then WRCM and RDCM are successively called, control returns to FLOOP where the cycle restarts.

3.1 Main Processing Loop Diagrams

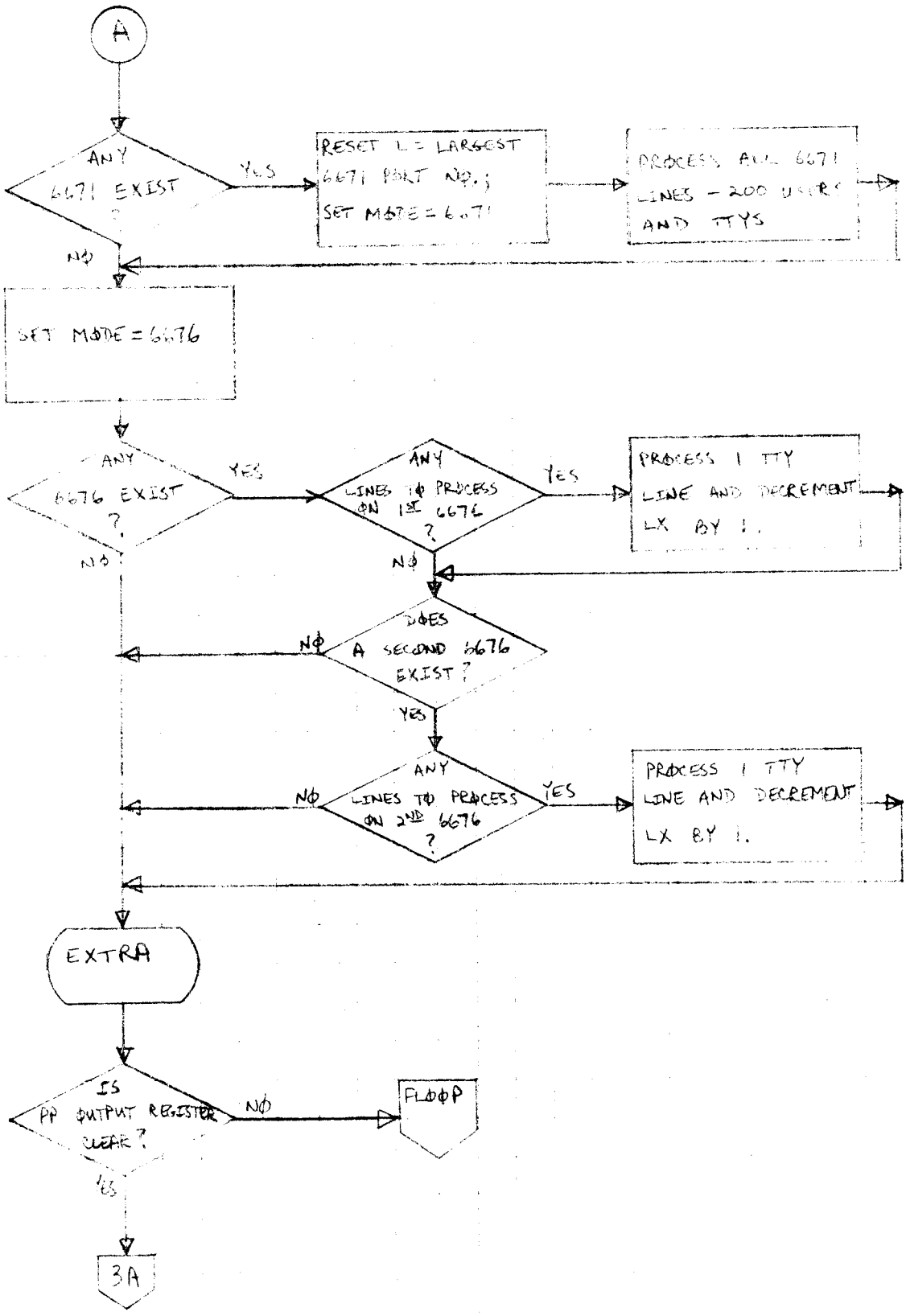
I/O DRIVER MAIN FLOW

PAGE 1



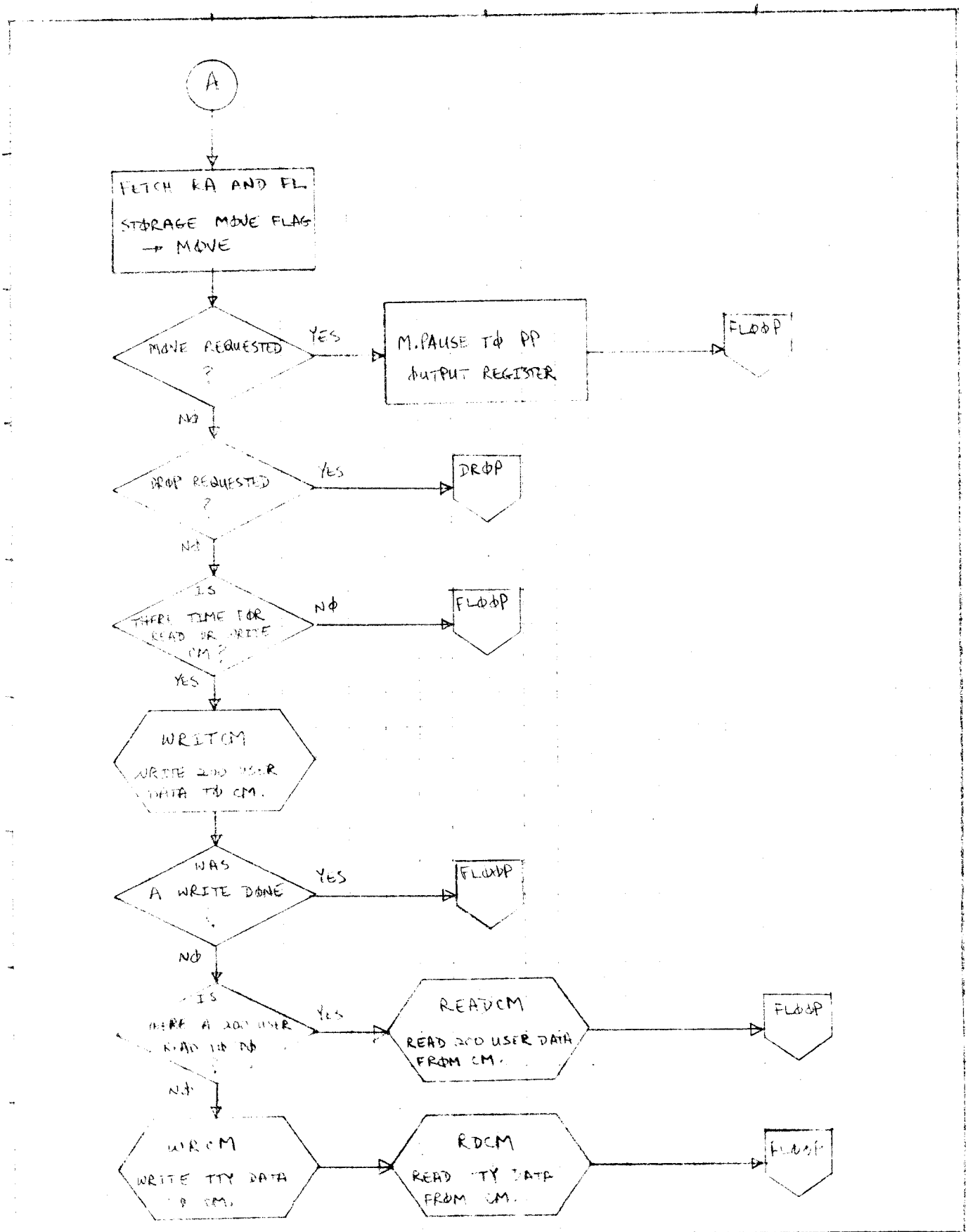
I/Ø DRIVER
MAIN FLØW

PAGE 2.



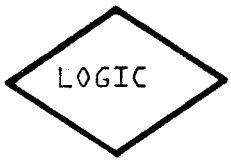
I/Ø DRIVER
MAIN FLOW

PAGE 3



4.0 KEY TO TASK FLOW DIAGRAMS

Task is the label of the beginning of a sequence of code which represents an input or output processing task. This location would be stored in ITASK for input processing or in OTASK for output processing. Control goes to this sequence only from the main line sequencing loops at LOOP {200 USER} or TLOOP {TTY}, as represented by the small circles {see below}. Each task is entered only once during each of the respective multiplexors cycles.

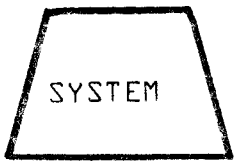


Logic is a decision box necessary to the changing of a task. Each task may contain addition logic which is not shown, because it doesn't affect the task flow. If two tasks are directly connected without logic, then flow will be unconditional.

Queue is a place where an I/O request is queued for a system routine.

Other represents code which resets the other task flow. If input task flow is being diagrammed then a change to OTASK would be in other.

Subroutine is a closed sequence of code which may change ITASK or 0TASK but from which control directly returns to the current task which called it.



System is a closed sequence of code which is never called from a current task, but which may reset ITASK or 0TASK, the logic shown under the system box is contained in that subroutine and is shown only to the extent necessary to show how the task may be reset.

Solid lines represent the direct control of task sequencing by the tasks themselves.

Dashed lines represent the end of the direct control of task sequencing by the tasks themselves. These lines will always connect to a system subroutine.

The small circle represents the break in execution where control is returned to the driver executive to process a task for another line. If the line from the small circle connects to a new task then it is assumed that ITASK or 0TASK was reset prior to the break.

5.0 OUTPUT TASK FLOW NARRATIVE - 200 USER TERMINAL

SCAN:

SCAN attempts to activate the line. If the storage move flag (MOVE) is set, SCAN exits immediately. Otherwise, the Line Status (LSW) and Line Request Words(LRW) are fetched (via RLSW) and PLB is tested for party line status. When PLB is non-zero, the LWSA is reset and RLSW is reentered to obtain the LSW and LRW of the new line. The terminal lock bit (TL) and error processing bit (EP) in the line status byte (LSB) are checked at SC0 and SCAN exits to TLOOPE when either TL=1 or EP=1. Otherwise (TL=0 and EP=0), SD and RD in the line request byte (LRB) are tested for zero at SC2. If they are both zero, then the subsystem is requesting the driver to set TL=1. Control goes to SC25 where the reply status byte (RSB) is cleared, CC, CS and IP in the line status byte (LSB) are cleared, TL is set to 1 and the LSW is written back to CM. SCAN then exits to TLOOPE.

When SD and RD are not both zero, then the subsystem has requested a transmission. The in process flag (IP) is checked at SC22 and if set (IP=1), control passes to SC40. Else, CC and CS are cleared in LSB and SD is checked for a WRITE request (SD≠0) at SC3. If SD≠0, then IP is set at SC4 and the LSW is written to CM and SCAN exits.

This is done only to break up SCAN into a small enough execution time. At SC3, if SD=0, then RD must be non-zero, and both IP and CC are set in LSB at SC5 and SCAN exits as above.

At SC40, the terminal ready and carrier on bits (IN) are checked to see if the data set is still operating. If neither of these status bits are set, then EC=7 is set in RSB to signal a timeout error and the telephone connect code (7000_8) is placed in OUT. Terminal lock is set (TL=1), the LSW is written back to CM, and SCAN exits to TLOOPE. If the data is OK, then ID in LSB is checked at SC41 for assignment to RESPOND and if so, DR is fetched from the previous reply status byte and inserted at CR in LRB. In any case, LRB is placed into the WS table at SC50 for use by later tasks. CC from LSB and the site address (SA) and current sequence bit (SB) from TSB are combined with a sync count of four to generate a value for the LS table. After SCAN, no later OTASK needs to reference CM, because all the necessary information to generate the output message is stored in the WS and LS tables.

Next, CC is checked and if set (indicating a POLL message), control skips ahead to SC60. Otherwise (CC=0), MT is checked and when MT is 3 or larger, indicating a WRITE message, the write acknowledged bit (WA) is cleared in TSB.

At SC60, ITASK is set to NT (which causes any spurious input data to be ignored) and CS is incremented to one. WAIT is set with MTAR to allow for the line turnaround to complete. Finally, OTASK is reset to SSYNC before SCAN exits at SC100 to TLOOPE. SCAN assumes that OUT contains the 200 USER sync code (4026).

SSYNC:

SSYNC delays advancing the OTASK sequence until at least 4 sync words have been sent out. Each entry causes the low order 3 bits of LS to be decremented until the count goes to zero. When not zero, SSYNC exits directly to NEXTL, where WAIT is reset. When zero, CS is incremented to two and

the start of message code {SOM=4001} is stored in OUT, OTASK is reset to SSOM, and an exit is made to NEXTL.

SSOM:

SSOM increments CS to 3 and generates the site address using the SA value from LS as an index into the table SITE from which it stores a value into OUT. OTASK is reset to SSTAT.

SSTAT:

SSTAT generates the station address code and stores it into OUT. CC in LS is checked and if a POLL is indicated {CC=1} the value STAT {4340} is used and OTASK is reset to SPOLL. Otherwise {CC=0}, if MT=2 {alert}, the non-POLL station address {4141} is used. For WRITE messages, the sequence bit {SB} is used from LS as the 2^4 bit in the station address. LS is cleared and OTASK is reset to SCNTL.

SCNTL:

SCNTL obtains MT from the WS table and uses it as an index into the MESS table to obtain the proper control code to be placed into OUT. MT is checked to see if the WRITE requires delay syncs sent out after the control code. If so, {MT≥4}, seven is placed in LS as a count, OTASK is reset to SDA and SCNTL exits. Otherwise {MT≤3}, CR is checked at SCN3, and if set, a carriage return must be output prior to any data. OTASK is reset to SESCA and SCNTL exits. When CR=0, an attempt is made to queue a read request for READCM at SCN42. Although execution does not break before going to SCN42, the description is below, since SCN42 can be an OTASK on other occasions.

SDA:

SDA delays advancing the OTASK sequence until 7 sync words have been sent out after the control code. LS is decremented each entry and a SYNC code (4026) is stored in OUT prior to SDA's exit to NEXTL. When LS finally reaches zero control passes directly to SCN42 without a break. {See below}

SESCA:

SESCA places an escape code {4076} into OUT and resets OTASK to SCRRA.

SCRRA:

SCRRA places the carriage return code {4301} into OUT and passes directly to SCN42 without a break.

SCN42:

SCN42 attempts to queue a request for the reading of data from CM by READCM. JUNGCT contains an indicator of the status of the queue. If JUNGCT is non-zero, an earlier request is still pending and control passes to SCN43. Otherwise, JUNGCT is set non-zero and the line number is placed in L1. And then at SCN43, OTASK is reset to SDB and the task exits to NEXT1.

SDB:

SDB sends Sync codes to the terminal until READCM can process this line. A SYNC code is stored in OUT and control passes to SCN42 without a break.

READCM:

READCM is a system subroutine which is called from the driver executive at EXT2 whenever JUNGCT is non-zero. Using the line number contained in L1 READCM gets the next block of data from CM, and places it in a POOL buffer. The FWA-1 of the buffer is stored in BUFOUT and the LWA is stored in BUFLM. Each PP word in the buffer has two output characters. Before exiting, OTASK is reset to SLC.

SLC:

SLC increments BUFOUT and extracts the left character from the current pair of characters in the buffer. This 6 bit Display coded character is used as an index into the display to 200 USER BCD conversion table DISDCP. Entries in the table are 12 bits including the multiplexor control bits (4 high order) and the 200 USER BCD code with parity (8 low order). The converted value is stored directly into OUT at SLC1. The DISDCP table contains the 200 USER escape code (4076_8) as the first entry, so that a left character of zero, indicating the escape character in SCOPE display code, will cause the 200 USER escape code to be placed in OUT. After the converted value is stored in OUT above, a test is made for the escape. If not an escape, then OTASK is reset to SRC at SLC00 and SLC exits. If an escape was stored then at SLC4, the 12 bit value in the POOL buffer is checked for special codes. If zero, signifying end of line (EOL) or carriage return (CR), then control goes back to SLC00 where OTASK is reset to SRC. If the 12 bit value is greater than 0026_8 then the code is invalid and the zero left character is assumed to be a 55_8 (blank). A 200 USER blank (DCPBLK) is stored in OUT and control goes to SLC00. If the 12 bit code is 0026_8, which signifies a data compression code , then OTASK is reset to SCC and SLC exits. For the end of data code ($EOD=0001_8$) and end of buffer code ($EOB=0002_8$), a sync word (4026_8) is placed in OUT at SLC10 and OTASK is reset to SLC20 before SLC exits.

SLC20:

SLC20 obtains the address of the current pair from BUFOUT and checks the 12 bit value at that location for the EOB code. If not, then EOD is assumed and RETBUF is called with the A-register zero, to return the POOL buffer, then OTASK is reset to SDT at SLC24 and SLC20 exits. If the value was an EOB code, the control goes to EJUNG where RETBUF is called with the A-register zero. The SD field in the WS table is next checked to determine the output buffer type. If SD=3 (message buffer), then control goes to SLC24, where OTASK is reset to SDT and SLC20 exits. Otherwise, for SD=1 or 2, OTASK is reset to SCN42 at EJ10 and SLC20 exits.

SEOB:

SEOB places a 200 USER SYNC word into OUT and then control passes to EJUNG (See SLC20 above).

SCC:

SCC uses the 12-bit compression code generated by the subsystem as an index into the table COMPCD. The table value is the appropriate 200 USER code for compressed zeroes or blanks. The address contained in BUFOUT is checked against the LWA of the buffer in BUFLM and the indication is stored in LAST. Control goes to SRC2 where the 200 USER code is stored in OUT, etc.

SRC:

SRC obtains the current word address in the POOL buffer from BUFOUT and compares it with the LWA in BUFLM. The end of buffer indication is stored in LAST to be tested later. The current 12 bit data item is fetched and checked for zero. If zero, then an EOL or CR is signaled and control goes to SRC05 where DT is checked. If DT=2 indicating that the data was sent to the printer, then the 200 USER end of line code (EOL=4320_8) is stored in OUT

at SRC2 and the end of buffer indicator in LAST is tested. If LAST is not-zero, indicating not at end of buffer, then control goes to SRC 15 where OTASK is reset to SLC and SRC exits. If LAST is zero (end of buffer) then OTASK is reset to SEOB and SRC exits.

If the current 12 bit data item is not zero, then the right character is obtained and used as an index into the DISDCP table to fetch the 200 USER BCD code to be placed into OUT. A check is made to see if the right character was an escape. If not, control passes to SRC2 (see above description). If zero, then a no data is assumed. (i.e., an odd number of characters were output) and a SYNC word is stored in OUT at SRC2. (See above).

SDT:

SDT checks DT in the WS table to determine if an E-code should be output to the 200 USER terminal. If DT=0, then control goes directly to SETX. Otherwise, an escape code (4076_8) is placed in OUT, OTASK is reset to SNDT and SDT exits.

SNDT:

SNDT obtains DT from the WS table and uses it as an index into the DT table to obtain the proper BCD code for the 200 USER E-code. This value is placed into OUT at SND10. OTASK is reset to SETX and SNDT exits.

SPOLL:

SPOLL stores the POLL control code {POLL=4205_8} into OUT at SND1.0, resets OTASK to SETX and exits.

SETX:

SETX clears the entry in table LS, so that the RSB can be accumulated during the input phase. The 200 USER ETX code {ETX=4203_8} is placed into OUT, OTASK is reset to SMPC and SETX exits.

SMPC:

SMPC functions the multiplexor to send the Message Parity character {MPC} and resync for the reply message from the terminal. a 3000_8 is placed into OUT and OTASK is reset to RESYNC before SMPC exits.

RESYNC:

When this output task is entered the multiplexor has already accepted the 3000_8 function setup by SMPC. RESYNC increments CS to 4, which indicates that the MPC has been output and the line turnaround has {or will shortly} started. OUT is cleared, so that the multiplexor will receive only NOP functions during input. The table WS is reformatted, by clearing the entry except for RD.

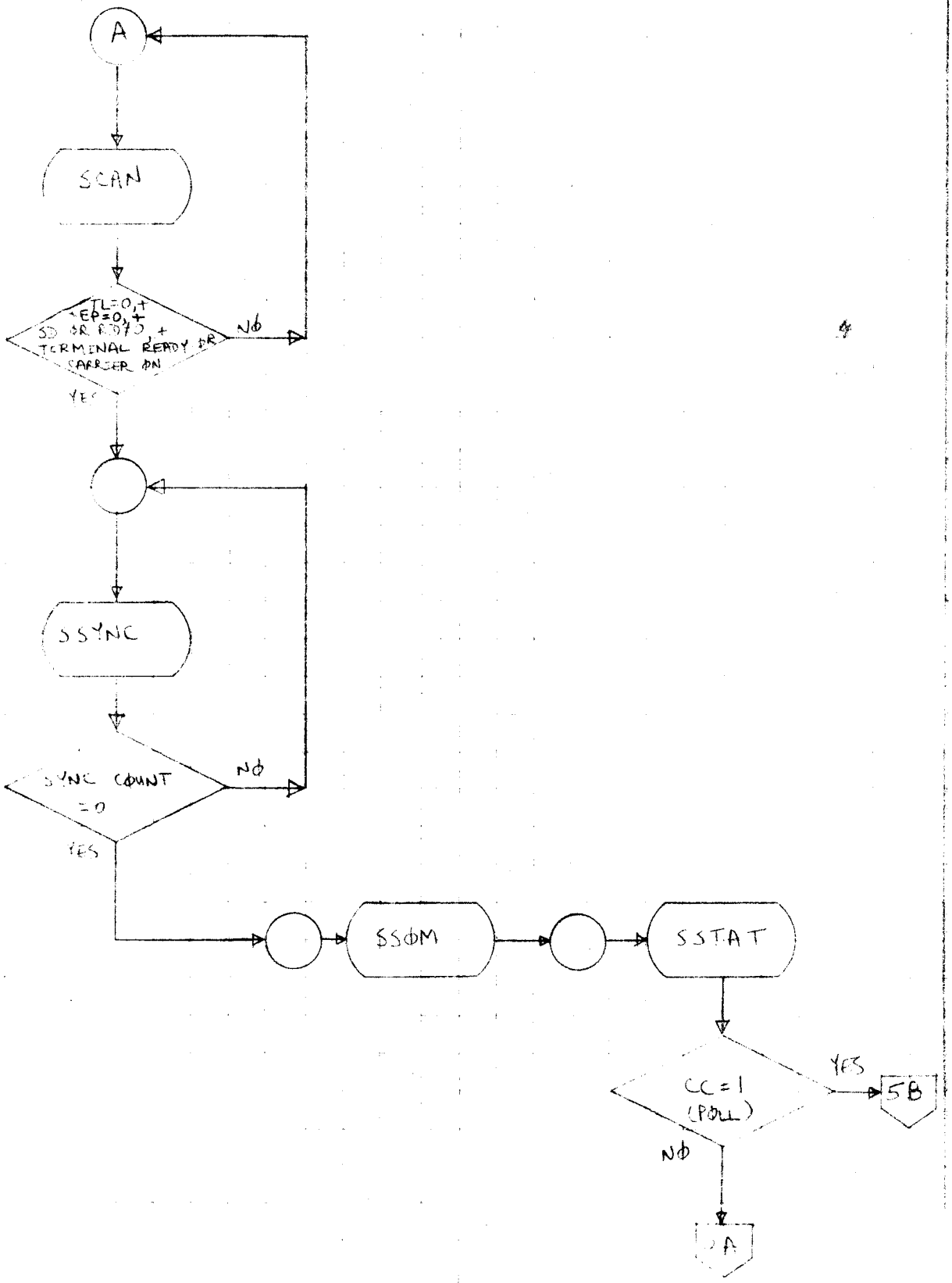
WAIT is set with MTAR, to allow for the line turnaround to be completed. Then OTASK is reset to REJECT. This provides for the millisecond timing of the input operations when no valid data character is received. Finally ITASK is reset with RSOM and RESYNC exits to NEXT. This completes the output task flow.

5.1 Output Task Flow Diagrams - 200 USER Terminal

I/Ø DRIVER

OUTPUT TASK FLOW DIAGRAM

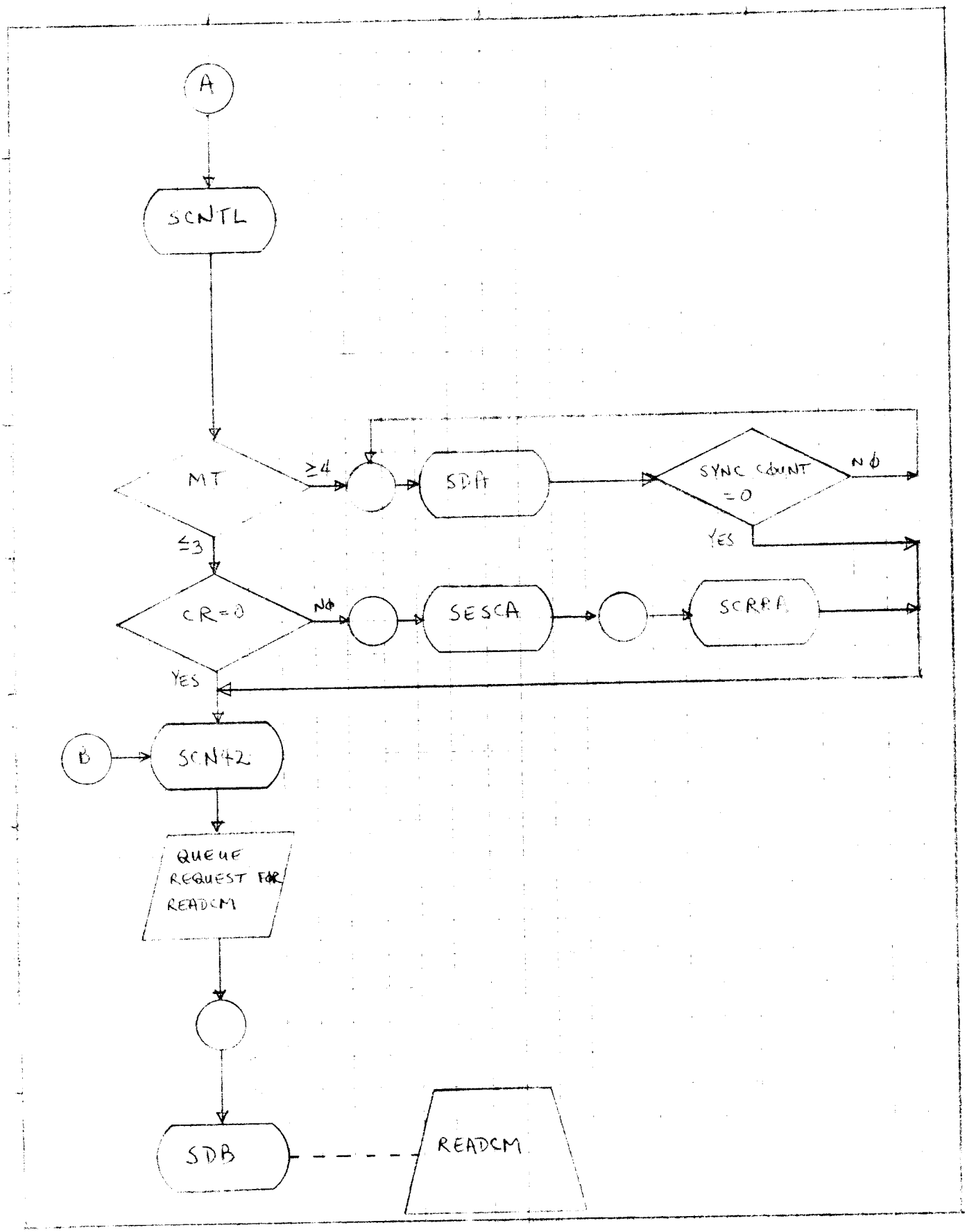
200 USER TERMINALS PAGE 1



I/O DRIVER

BUILT TASK FLOW DIAGRAM

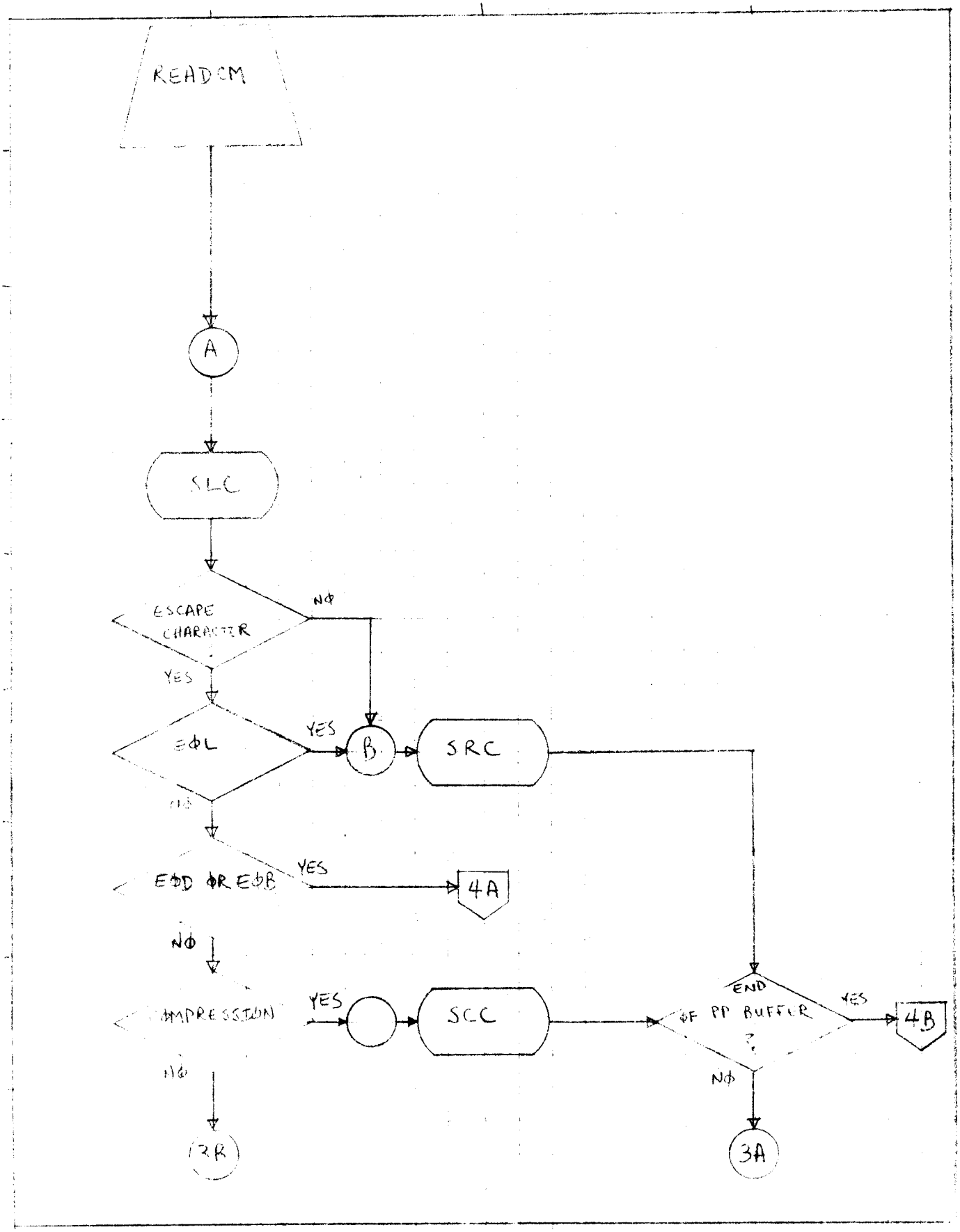
200 USER TERMINALS PAGE 2



I/O DRIVER

OUTPUT TASK FLOW DIAGRAM

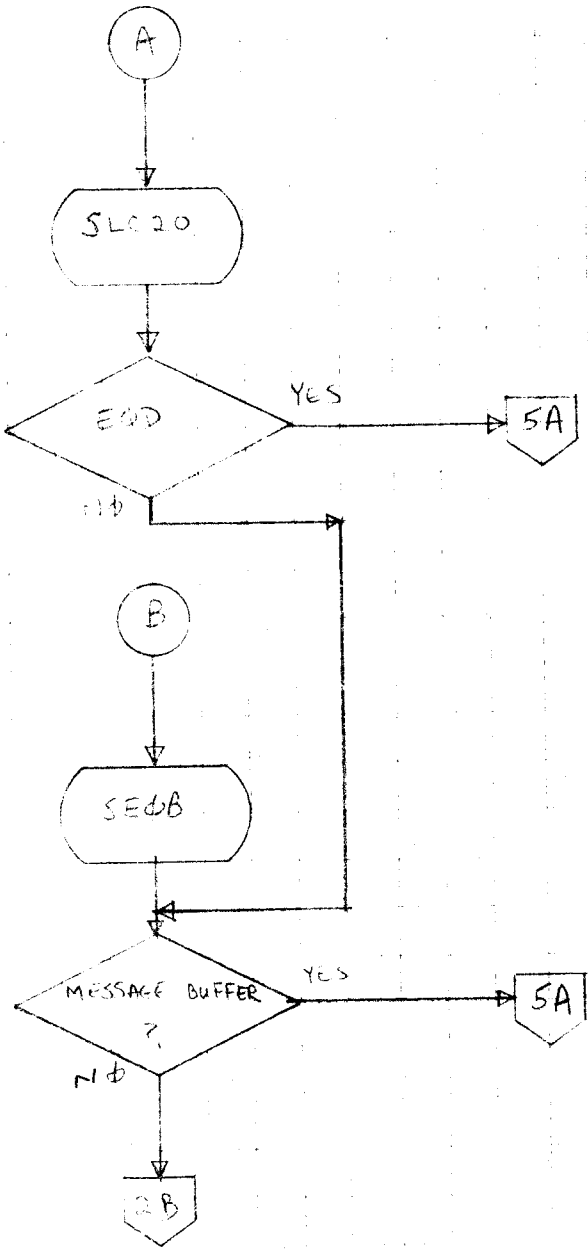
200 USER TERMINALS PAGE 3



I/O DRIVER

OUTPUT TASK FLOW DIAGRAM

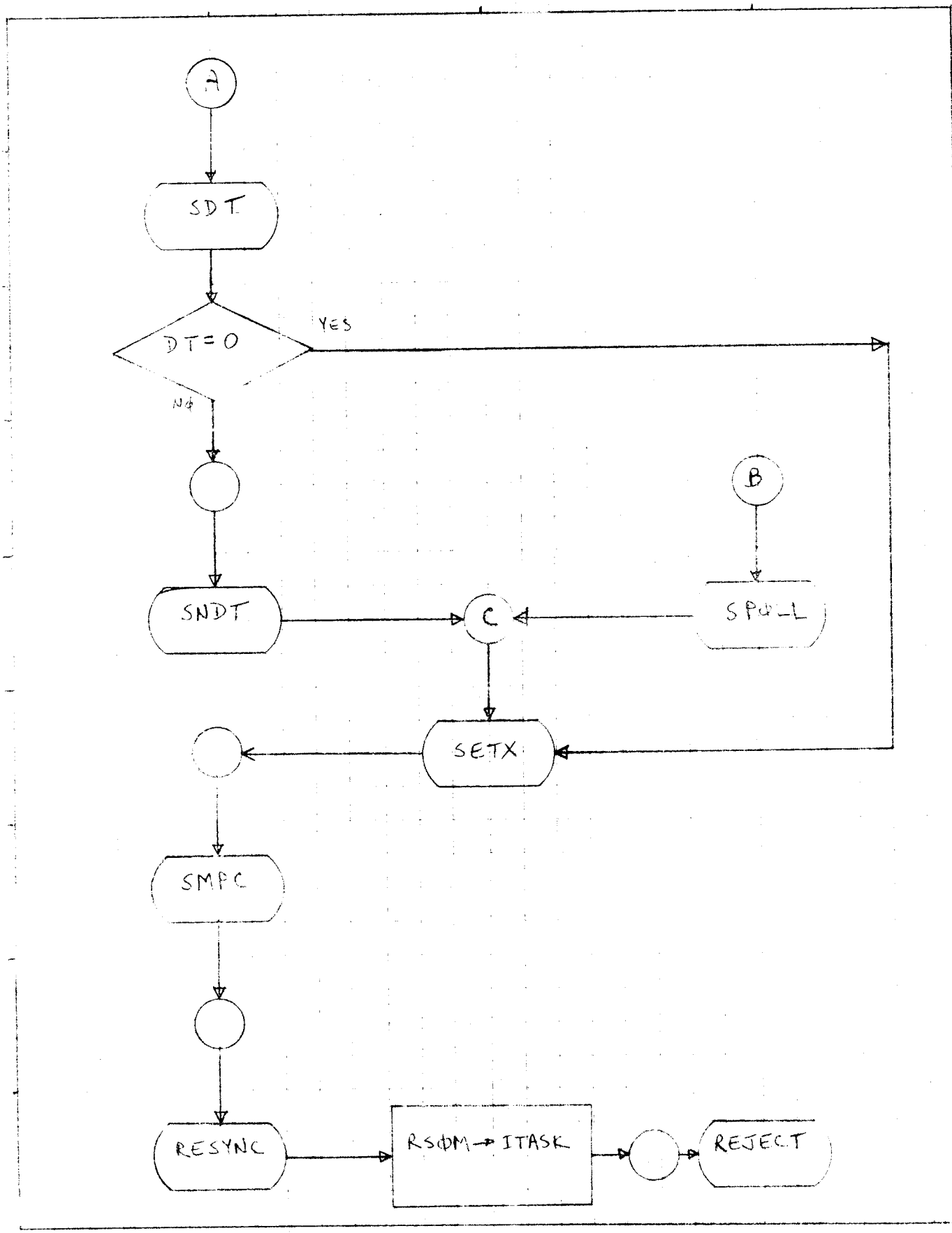
200 USER TERMINALS PAGE 4



I/O DRIVER

OUTPUT TASK FLOW DIAGRAM

200 USER TERMINALS PAGE 5



6.0 INPUT TASK FLOW NARRATIVE - 200 USER TERMINAL

When the main line sequencing executive at LOOP detects a °character ready° status for the line, control passes to VAL where the 8 bits of data are saved in TEMP. If the data item is not an ETX {203_8} then WAIT is reset with the maximum character timeout {CHWAIT} and the address of the current input task is obtained from the table ITASK. Control goes to that task with the contents of TEMP in the A-register. When an ETX is received see PETX below. Following is a description of each input task.

RSOM:

RSOM checks that the first character received is the SOM {001_8}. If it is not a SOM, then control passes to BADC, where an error code {EC} of 3 is stored into the table LS {where the RSB is accumulated} and ITASK is reset to NEXT. RSOM then exits to NEXT. Setting ITASK to NEXT creates a null task, which effectively throws away all the data {or garbage} received until an ETX appears or a timeout appears.

However, if a SOM is received as the first character then RSOM increments CS to 5 {which indicates that the input phase has started}. ITASK is reset to RSITE and RSOM exits to NEXT.

RSITE:

RSITE checks the next character for the $16X_8$ or $17X_8$ value indicating a site address code. If not, then control goes to BADC, else ITASK is reset to RSTAT and RSITE exits to NEXT.

RSTAT:

RSTAT checks the next character for a 140_8, 141_8, 160_8 or 161_8 station address code. If not, control goes to BADC, else the message sequence bit is fetched from the 2^4 bit of the station address and is saved in the table LS as SR. ITASK is reset to RCNTL and RSTAT exits to NEXT.

RCNTL:

RCNTL checks the next character for a valid message control code. First a check is made for the READ control code (023_8) and if found, control goes to RCN1 where RC=1 is stored into the table LS. ITASK is reset to RCLB and RCNTL exits to NEXT. If the control code was not a READ, then checks are made for ERR (025_8), ACK (206_8) and REJ (230_8). If any of the three are found then 3, 4 or 2 respectively is stored as the RC value in the table LS and ITASK is reset to NEXT before RCNTL exits. The READ message is the only one which does not have an ETX character immediately following the control code. If one of the four valid control codes is not found then control goes to BADC. (See RSOM above). The data received after a READ control is converted to 6000 Display code and stored in pairs (two 6 bit characters per 12 bit PP word) by the following three tasks.

RCLB:

RCLB is the task which receives the first data character for a new buffer. RCLB checks the character and if it is an escape, goes to RCLD2 where ITASK is set to RCP. Otherwise, when the character is not an

escape, RCLB calls subroutine GETBUF to obtain a POOL buffer. The FWA of the new buffer is stored in FROM and the table BUFIN. At RCLB25 the current character which is stored in TEMP, is used as an index into the BCD to Display conversion table (CONV) to obtain the first display data character for this new buffer. The value from the CONV table is checked for a parity error indication. If the value is zero, or has the high order bit on, then a parity error was detected and SETEC4 is entered to set EC=4. In either case, ITASK is reset to RCRA at RCLB3 and RCLB exits to NEXT.

RCRA:

RCRA is the task which receives a data character to be stored in the right side of the current PP word in the POOL buffer. RCRA checks the character and if it is an escape, goes to RCLD2, as in RCLB above. Otherwise, the current location in the POOL buffer is fetched from BUFIN and compared with the LWA in BUFLM. The end of buffer indication is saved in direct cell D.Z2. The current character is converted to Display code by using TEMP as an index into the CONV table. If a parity error is indicated, then SETEC4 is called from RCRA3. Otherwise, the converted character is stored as the right character and the end of buffer indicator is tested at RCRA4. If not the end of buffer, ITASK is reset to RCLD and RCRA exits. At the end of buffer, BUFIN is cleared at RCRA6 and QCMW is entered with the A register clear to queue a request for WRITCM. Then control goes to RCN2 where ITASK is reset to RCLB and RCRA exits.

RCLD:

RCLD is the task which receives a character to be stored on the left side of the next PP word in the POOL buffer. RCLD checks the character, and if it is an escape, goes to RCLD2 as above. Otherwise, the current address in the POOL buffer in BUFIN is incremented to the next PP word and saved in FROM. Control goes to RCLB25. (See description in RCLB).

RCP:

RCP is the task which processes the character after an escape character. First, ITASK is reset to the original task which detected the escape character. Then the routine RCESP is executed.

RCESP:

RCESP is the part of the task RCP which processes the character after an escape character. RCESP tries to determine which escape code sequence has been received. RCESP checks the character for a carriage return ($CR=301_8$), and if it is, then control goes to IGNORE where the task exits to NEXT. If not CR, then a check is made for end-of-file ($EOF=326_8$), and if it is, a 40_8 code is saved in D.Z2 and control goes to RCLC15. At RCLC15, BUFIN is tested for zero, which would indicate the EOF or EOR code appeared in column 1 of the card. If not zero, then EC=2 (Bad Card Code) and DS=3 are set into LS at RCLC7 and RCESP exits to NEXT. If BUFIN is zero, then GETBUF is called from RCEO and the EOR or EOF code which was saved in D.Z2 is stored in the first PP word of the new POOL buffer. ITASK is reset to RCLEV and RCESP exits to NEXT.

If the character being processed by RCESP is not an EOF, then a check is made for end of record (EOR=127_8), and if it is, a 00_8 code is saved in D.Z2 and control goes to RCLC15. (see above) If not EOR, then a check is made for an E1 (E1=302_8), E2 (E2=040_8), or E3 (E3=241_8).

If no E code is detected, then EC=2 (Bad Card Code) is set into LS at RCLC7 before an exit is made to NEXT. For an E2 or E3, BUFIN is first tested for zero, and if not, control goes to RCLC7 to set EC=2. Otherwise, a 1, 2, or 3 is inserted for the respective E code into the DS field in the table LS. If BUFIN is zero, then the "queue-null-buffer" flag is set in WS at RCLC45 and control goes to RCLC40 where ITASK is reset to NEXT. (After an E code, the next character received should be an ETX.) Otherwise, BUFIN is compared with BUFLM as an end of buffer test. If they are not equal, then an EOD code (0001_8) is inserted into the POOL buffer. BUFIN is cleared at RCLC39 and the "queue-last-buffer" flag is set in WS at RCLC39, and RCESP exits through RCLC40 as above.

RCLEV:

RCLEV processes the character after the EOR code, which is the high order digit of the record level number. A binary representation of that digit (0 or 1) is inserted into the low order 3 bits of the first word in the POOL buffer. Control goes to RCRA2 where ITASK is switched to RCLD and RCLEV exits to NEXT.

PETX:

PETX is main line code reached from VAL when a new character is received and VAL detects that the 8 bit character is the end of text (ETX=203_8). Every character received is checked for ETX, so that the missing or no E code case is not flagged as a timeout error. If the only error is the missing E code, then the error code is set to zero, but DS=0.

PETX checks the 2^2 bit of CS to determine the status of the communication. When CS$\leq$3, control goes immediately to NT and an input is ignored. For CS$\geq$4, the "ETX-received" flag in WS is checked, and, if set, control goes to VAL5. Otherwise, the "ETX-received" flag is set. This is done in case the message parity character (which is zero to indicate no error) happens to be equal to the ETX character (=203_8). QCMW is called if the "queue-last-buffer" flag in WS is set. RELBUF is called in case a stray POOL buffer was not queued to WRITCM yet. Then ITASK is reset to RMPC and PETX exits to NEXT.

RMPC:

RMPC is the task which checks the character immediately after the ETX. If the character is non-zero, then a message parity character error occurs and EC=5 is inserted into the table LS, unless some previous error has already been noted. If there is no MPC error, or after the EC is set, CS is incremented to 6, ITASK is set to NT, OTASK is set to RMP1, and RMPC exits to NEXT.

RMP1:

RMP1 is the task which waits for the last buffer to be written to CM by WRITCM. First, a SYNC code is placed in OUT to start the line turn-around. Then, the "last-buffer-not-written" flag (BN) in WS is checked. If set,

RMP1 exits to NEXT. This check is made to prohibit the setting of terminal lock (TL=1) until all of the received data is written to CM. If the flag is not set, the value of CS is saved in WS, CS is cleared to ignore any input characters at LOOP, OTASK is set to RMP10, and RMP1 exist to NEXT.

RMP10:

RMP10 through RM10 are the tasks which check the error code (EC) and the response code (RC) and update the LSW and LRW when necessary. This has been divided into a number of small tasks because of the timing requirements for running 32 200 USER Terminals.

RMP10 is the task which checks the error code (EC) to determine of some error recovery procedure is necessary. First, the move flag is tested. If set, (move in process), RMP10 exits to NEXT. Otherwise, the LSW is read from CM, the 3 bits of CS in WS are inserted into the LSB and saved in WS, and the TSB is saved in ITASK. (Note that ITASK is not used when CS=0.) At this point, the LSB (in WS), TSB (in ITASK), and RSB (in LS), are all available in PP memory, thus eliminating the necessity of reading the LSW from CM for these bytes in subsequent tasks.

Next, the EC in LS is checked. If EC$\geq$3, the subsystem ID in WS is checked. If the line is unassigned (ID=0), then OTASK is set to RMP15. If ID$\neq$0, then OTASK is set to RM7. In either case, RMP10 exits to NEXT. If EC$\leq$2, then no transmission error occurred in the response from the terminal. At RM73, the write acknowledged bit (WA) is tested. OTASK is set to RM6 if WA=1, or RM73A if WA=0, and RMP10 exits to NEXT.

RMP15:

RMP15 handles the error recovery for unassigned lines. If CS≠4, OTASK is set to RM7 and RMP15 exits to NEXT. If CS=4, the move is checked. If set, RMP15 exits immediately to NEXT. Otherwise, the LSW is read from CM and the PLB is checked. If this is a party line (PLB≠0), then OTASK is set to RM7 and RMP15 exits to NEXT. If the line is not a party line (PLB=0), the site address is incremented and saved in ITASK, OTASK is set to RM8, and RMP15 exits to NEXT.

RM73A:

RM73A compares the sequence bit sent (SB in ITASK) with the sequence bit received (SR in LS), and if they are the same, sets WA=1, complements SB, sets OTASK to RM6, and exits to NEXT. If the sequence bits are not the same, then the WRITE message was not received correctly by the terminal and CC in WS must be checked to determine the proper error recovery procedure. If CC=0, the last message sent to the terminal was a WRITE; hence, the terminal must be POLLed to re-position the entry marker before the WRITE can be re-transmitted. With the entry marker re-positioned, the second WRITE will overwrite the one in error, and the screen will not show the transmission error. To POLL the terminal, control goes to RM7 (see below). If CC=1, the recovery procedure is to perform the WRITE again. (i.e., after a POLL to re-position is completed and CC≤2, then CC=1 and a reWRITE is necessary). To re-transmit the WRITE, CC is cleared, TL and IP are set to 1, and control goes to RM9 (see below at RM8).

RM6:

RM6 is executed if the last transmission was correctly received by the terminal (WA=1). First, the move flag is tested, and if set, RM6 exits to NEXT. Otherwise, the LRW is read from CM and SD and CR in the LRB are cleared. Then MT is checked for a single shot POLL (MT=1), and if so, OTASK is set to RM77 where TL is set and IP is cleared. If MT≠1, OTASK is set to RM6C. In either case, the LRW is written back to CM and RM6 exits to NEXT.

RM6C:

RM6C clears TL and IP in WS and checks the response code (RC in LS). If RC=1 (READ), control goes to RM8 (see below). If RC≠3 (ERR), OTASK is set to RM10 and RM6C exits to NEXT. If RC=3 (ERR), control goes to RM7.

RM7:

RM7 sets TL, IP, and CC in WS to 1 (error recovery in process, rePOLL the terminal), and control goes to RM9 (see below at RM8).

RM77:

RM77 sets TL to 1 and clears IP after a single shot POLL (MT=1) has been correctly received by the terminal. Control then goes to RM9 (see below at RM8).

RM8:

RM8 sets TL to 1. Then, at RM9, a check is made for both TL=1 and IP=1. If either TL or IP is 0, or the error code (EC) is less than 3 or greater than 6, then control goes to RM9A where OTASK is set to RM10 and RM8 exits to NEXT. However, if both TL and IP are 1, and $3 \leq EC \leq 6$, then TL is cleared, EP is set to 1, and control goes to RM9A.

RM10:

RM10 is the task which writes the new LSW back to CM. First, the move flag is checked, and if set, RM10 exits immediately to NEXT. Then, if move is not set, the LSW is read from CM, WS is stored in LSB, ITASK in TSB, LS in RSB, and the LSW is written back to CM. OTASK is set to RM20, and RM10 exits to NEXT.

RM20:

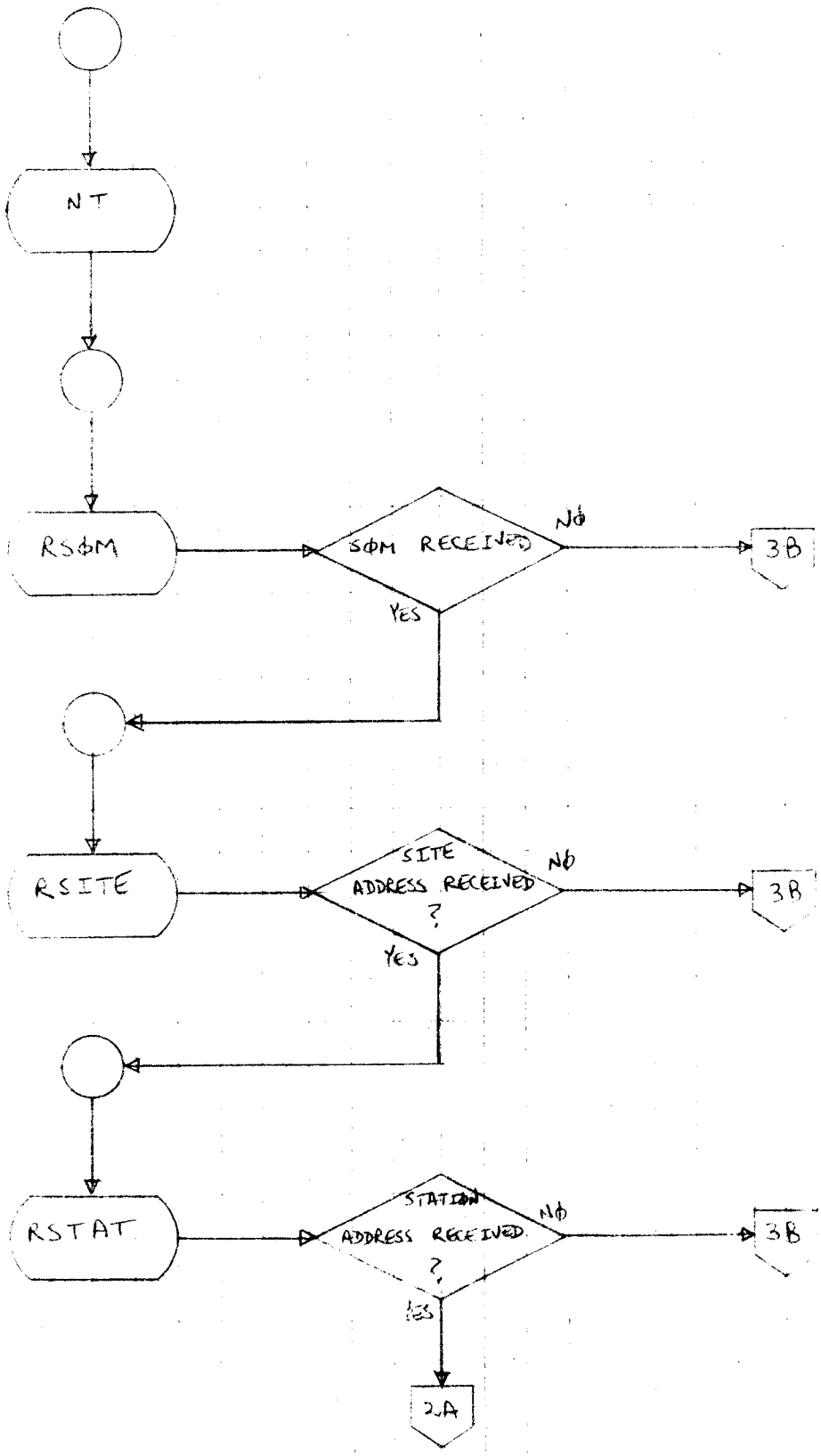
RM20 sets ITASK to NT and transfers control to INT1 where OTASK is set to SCAN and an exit is made to TLOOPE.

6.1 Input Task Flow Diagrams - 200 USER Terminal

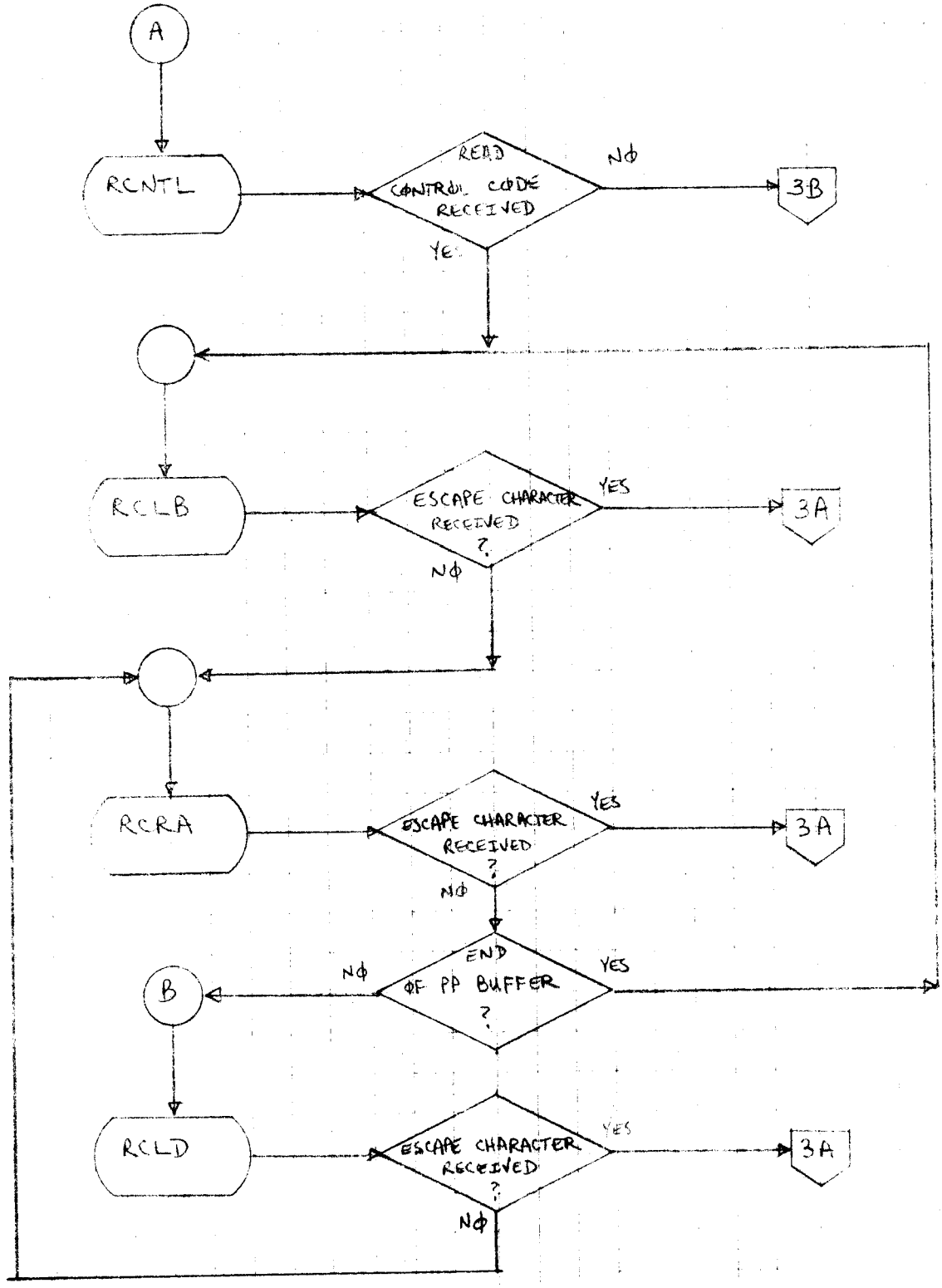
I/Ø DRIVER

INPUT TASK FLOW DIAGRAM

200 USER TERMINALS PAGE 1



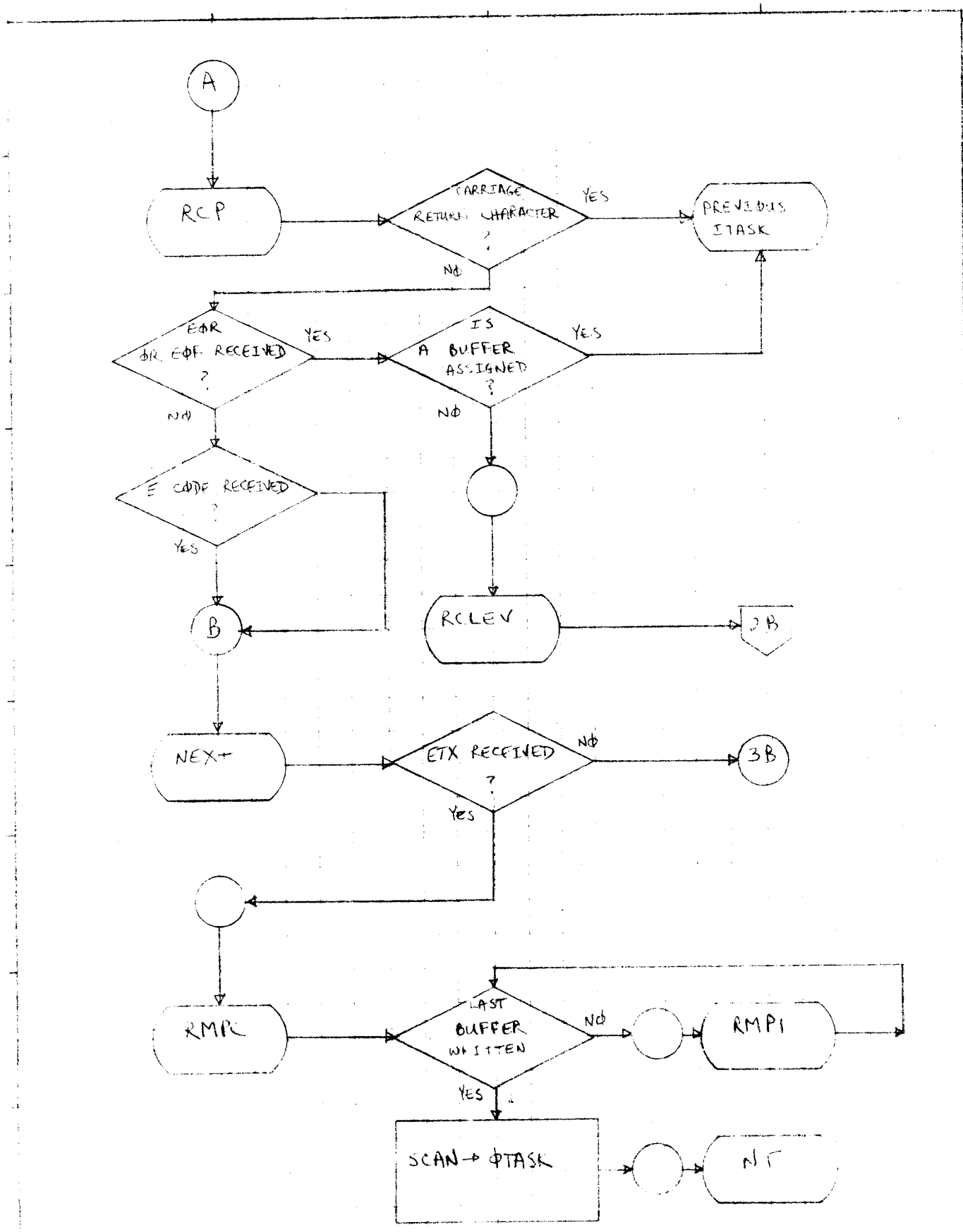
I/O DRIVER
INPUT TASK FLOW DIAGRAM
200 USER TERMINALS PAGE 2



I/Φ DRIVER

INPUT TASK FLOW DIAGRAM

300 USER TERMINALS PAGE 3



7.0 OUTPUT TASK FLOW NARRATIVE - TELETYPE TERMINAL

SCAN:

SCAN attempts to activate the line. If the storage move flag is set (MOVE $\neq$ 0), SCAN exits immediately to TLOOPE. Otherwise, the LSW is fetched from CM. Terminal lock (TL) and the Error Processing (EP) flag are checked in LSB and if TL=1 or EP=1, then SCAN exits to TLOOPE. Otherwise (TL=0), SD and RD are tested for zero at SC2. If they are both zero, the subsystem is requesting the driver to set TL=1. Control goes to SC25 where RSB is cleared, CC, CS and IP are cleared and SIDL is called to set TL=1, OTASK=SCAN and ITASK=TLOOPE. SCAN exits to TLOOPE.

When SD and RD are not both zero, then the subsystem wants the TTY line activated. IP is tested at SC22. When IP=1, control goes to SC40 (see below). If IP=0, then CC and CS are cleared in LSB and SD is tested at SC3. If SD=0, then IP and CC are set, else only IP is set in LSB. SCAN exits to TLOOPE to reenter the next cycle.

At SC40, the "terminal ready" status contained in IN is checked to determine if the TTY is still dialed up. If not, EC=7 is saved in the RSB and the line is reset to TL=1, OTASK=SCAN, ITASK=TLOOPE and control goes to TLOOPE.

At SC40B, PNB is checked for being a TTY line. Then OTASK is checked to see if the current task is SCAN1, and if so, SCAN1 exits to TLOOPE. If not, ITTY is called to activate the input side of the TTY. DT is checked in LRB for the paper tape mode of input. When DT=3, the paper tape flag is set in CS. If SD=0 (no output to the teletype), then ITASK is set to RTCH, OTASK is set to SCAN1, and SCAN exits to TLOOPE. If SD≠o, then ITASK is set to TBRK (test for break character), OTASK is set to TTYOUT, and SCAN exits to TLOOPE.

SCAN1:

SCAN1 is a TTY output task which is sort of an idling OTASK during the TTY input phase. The task is identical with the initial part of SCAN. The essential tests are:

1. When SD and RD=0, then set TL, and

2. When the "terminal ready" status drops, set TL.

In both cases, 0TASK is reset to SCAN and TL=1.

TTY0UT:

TTY0UT attempts to start output to the TTY. First TTY0UT checks M0VE and if non-zero, exits to TL00PE since CM cannot be read during a storage move. When M0VE=0, the LSW and LRW are fetched from CM and MT is checked for a pseudo I/0 transfer (MT=1). When MT≠1, control goes to TT14 below. For MT=1, subsystem zero has detected a L0GIN message in the message buffer and it must be transferred to CIRCSTACK for RESP0ND. A pool buffer is obtained via REQBUF (if none is available, TTY0UT exits to TL00PE) and 8 CM words are read from the message buffer into the P00L buffer. Calls are made to CSTACK to write the L0GIN command to CIRCSTACK. When completed the P00L buffer is returned via RETBUF and WBB is cleared for RESP0ND. SD is cleared, TL is set and 0TASK is reset to SCAN. The LSW and LRW are written to CM before TTY0UT exits to TL00PE.

When MT≠1, then TSB is initialized at TT14 with the current CM and byte positions in the buffer being read from CM. TSB=0003_8 implies CM position=000 and byte=3. Since RDCM can fetch only 1 byte (12 bits) from CM at a time, the current location in the CM buffer is kept in TSB. The current jungle unit number JBB is reset from WBB. If however, WBB=0, then RSB is cleared and the line is reset, SD=0, TL=1 and 0TASK is reset to SCAN. When WBB≠0, then a CM buffer is available and 0TASK is reset

to STRT. (A carriage return-line feed sequence is always sent at the beginning of each TTY output). The LRW and LSW are written to CM and control goes to STC4 below.

STRT:

This output task starts the carriage return-line feed output sequence. TEMP1 is set to STR1, the TTY carriage return character (4033_8) is set into OUT, and control goes to STC3 (see below).

STR1:

Subroutine QTYR, which queues a request for TTY data from CM, is called. QTYR also sets TEMP1 to TLOOPE so that no output is sent to the TTY until the CM read is done. The TTY line feed character (4025_8) is set into OUT and control goes to STC3.

RDCM:

RDCM is a system subroutine which is called from the driver executive near EXT2 whenever WRITCM or READCM are not called. RDCM checks the request queue length using JUIN and JUOUT. If they are equal, RDCM exits {empty queue}. Otherwise the line number of the request is fetched using the outpointer JUOUT and stored in L. If a break character has been received (ITASK≠TBRK), then output is suspended and RDCM exits. Otherwise the current CM and byte positions are obtained from TSB and the next pair of output data is fetched from CM and stored into the table TOUT. If the pair is a 12 bit zero {0000_8} then a carriage return-line feed sequence is required and OTASK is reset to STRT and RDCM exits. If not CR-LF, then TOUT is checked for the end of buffer code {EOB=0002_8}. JLINK is entered to link to the next jungle unit. If JLINK returns zero, then no data remains and control goes to RD45, see below. Else, TSB is reset {0003_8} and OTASK is reset to STRT to perform a CR-LF.

If TOUT is not EOB, then a check is made for end of data {EOD=0001_8}. If EOD, then at RD45, SD is cleared, ITASK is set to RTCH, OTASK is set to SCAN1 and RDCM exits. Finally if TOUT is not EOD, then more data exists and OTASK is reset to STCH before RDCM exits.

STCH:

TEMP1 is set to STC1. STCH then fetches the left-most 6 bits from TOUT, shifts it right 6 bits, and jumps to STC2 (see below).

STC1:

STC1 calls subroutine QTYR, fetches the low-order 6 bits from TOUT, and goes to STC2.

STC2:

The value in the A-register (either the right-adjusted upper 6 bits or the low-order 6 bits of TOUT) is left-shifted 17 bits and saved in TEMP, and the low-order 17 bits of the A-register are cleared. TEMP now contains the value of the display-coded character divided by 2 and the A-register has its sign bit on or off according to whether the display-coded character was odd or even. E.g., if the display-coded character is a B (=02), then TEMP=01, and the A-register is positive. If the display-coded character is a C (=03), then TEMP=01, and the A-register is negative.

TEMP is used as an index into the display to TTY conversion table, DISTTY. The left (right-adjusted) or right 6 bits of the table value are used according to whether the A-register is positive (even-valued display-coded character) or negative (odd-valued). This 6-bit character is multiplied by 2 and 4101_8 added to it to form the TTY character, the result is stored into OUT, and control goes to STC3.

STC3:

OTASK is set to the value saved in TEMP1 and control goes to STC4.

STC4:

STC4 checks the "terminal ready" and "carrier on" status bits in IN and if both are off, OTASK is set to SCAN1, ITASK is set to TLOOPE, and STC4 exits to TLOOPE. Otherwise, an immediate exit is made to TLOOPE.

WRCM:

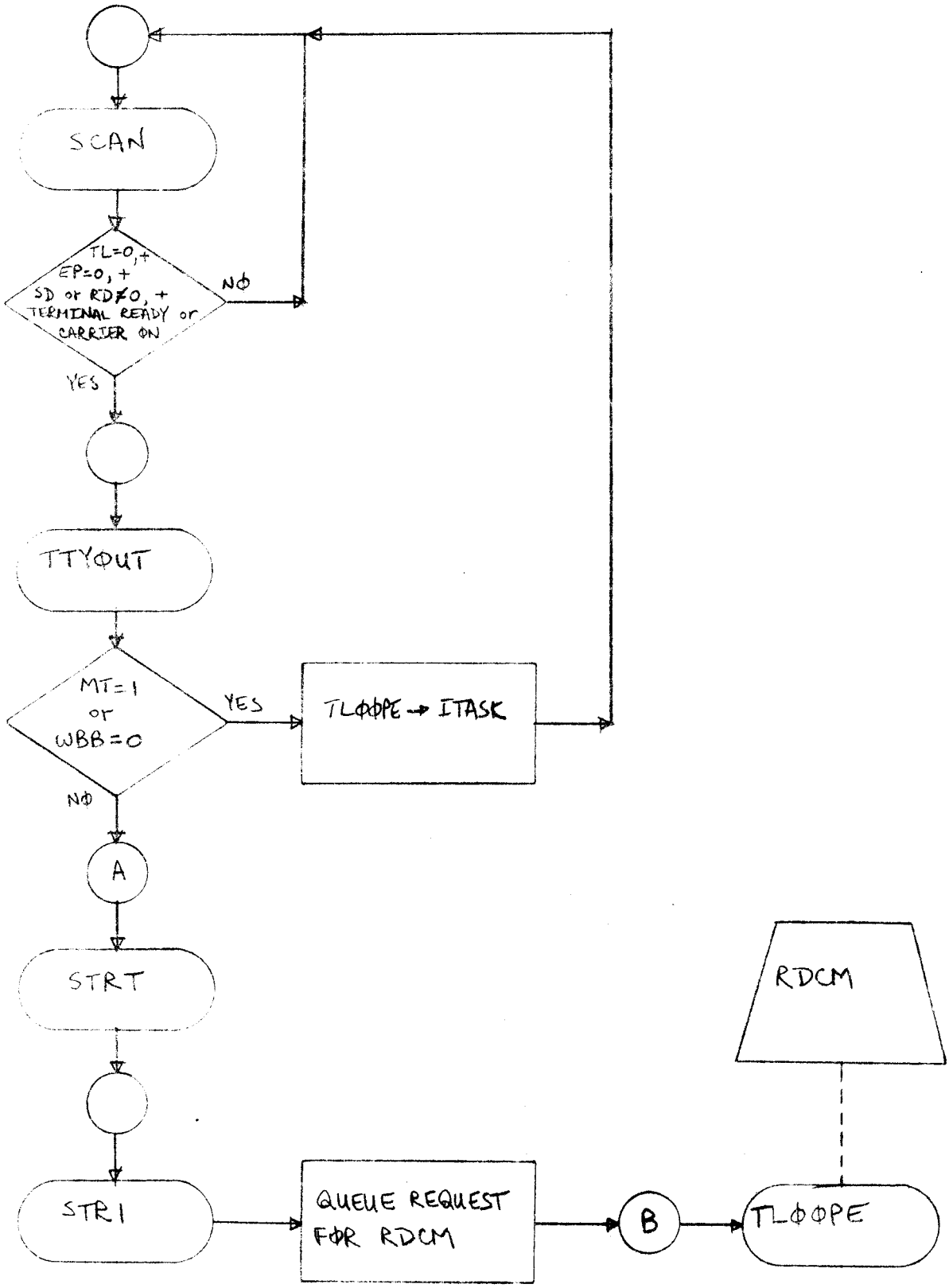
WRCM is a system subroutine which is called like RDCM. WRCM cycles between all the lines, processing one each time it is called. The current line number is contained in L2 which is decremented each entry to WRCM. When L2 goes to zero, it is reset to the value contained in LPP for the next entry. If TL=1 or the line is not a TTY, then WRCM exits immediately. If the TTY is not in the input mode, then WRCM exits immediately.

The rest of WRCM is devoted to writing TTY data to CM. When the EOD is finally written to CM control goes to WT50, where CS is tested for paper tape input mode. If set, then WRCM calls ITTY to reinitialize for TTY input, SINP to set ITASK=RTCH and OTASK=SCAN1, and exits. Else, SD is cleared, TL is set and a null character (4000_8) is placed into OUT. OTASK is reset to SCAN, ITASK is reset to TLOOPE, and WRCM exits.

7.1 Output Task Flow Diagram - Teletype Terminal

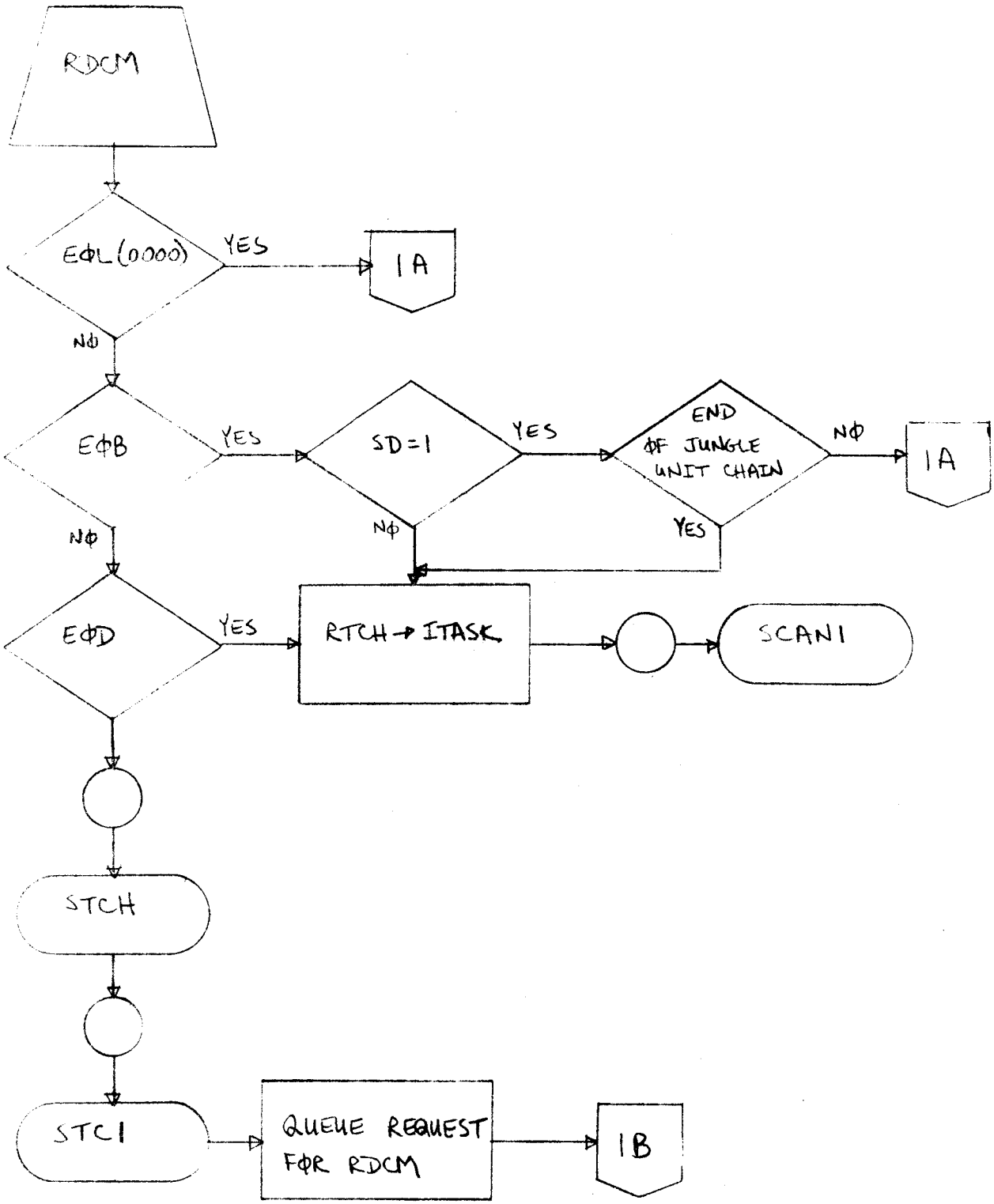
OUTPUT TASK FLOW DIAGRAM

TTY



ΦUTPUT TASK FLΦW DIAGRAM

TTY PAGE 2



8.0 INPUT TASK FLOW NARRATIVE - TELETYPE TERMINAL

Teletype input is enabled all the time when the terminal lock is not set (TL=0). Input is activated in the output task SCAN when the subroutine ITTY is called. When in the output mode (SD≠0), ITASK is set to TBRK to check for a break character. If SD=0, ITASK is set to RTCH.

Finally when the carriage return is received, further input is crippled, by setting ITASK= TLOOPE, until the current input line is completely processed to CM and TL is cleared again by the subsystem.

ITTY:

ITTY is a subroutine called from SCAN to activate TTY input. It is also called from the system routine WRCM after the EOD is written to CM when in the paper tape input mode. The second call is necessary since TL is not set after a carriage return is received in paper tape mode and SCAN would never be executed to call ITTY.

ITTY initializes CS with the TTY flag, the paper tape mode flag if DT=3, and the current position=right character. The break flag in CS is cleared. The input buffer pointer contained in BUFIN is reset to the FWA of the buffer. All TTY input buffers begin on an 8 PP word boundary. In addition, the buffer free code (FREE=0000) is stored in the fifth word of the buffer. Each 8 PP word TTY buffer is considered to be two 4 PP word buffers for processing data to CIRCSTACK. ITTY then exits.

SINP:

SINP is called by SCAN when SD=0 (output is complete, begin input), by WRCM after the EOD is written to CM when in the paper tape mode, and by TBRK when a break character is received.

SINP sets ITASK to RTCH (receive teletype character), OTASK to SCAN1, and exits.

TBRK:

The input task TBRK is executed when a valid character is received when in the output mode. The input character is checked for a break and if it is not, TBRK exits to TLOOPE. If the character is a break, SINP is called, the paper tape mode is cleared and the break flag is set in CS, and control goes to RTC4 below.

RTCH:

RTCH is the input task which processes the teletype input characters. Each character is converted to display code and stored in the PP buffer.

If TEMP (which contains the input character) is a tab character, the tab is replaced by a semi-colon for interpretation by the subsystem. A semi-colon (4167_8) is stored in TEMP1 and control goes to RTC3A below.

If TEMP contains a break character, the paper-tape mode flag in CS is cleared and control goes to RTC4 below.

If TEMP contains a carriage-return character, SDCH is called to get the current location in the PP buffer. An EOD is then stored into the buffer, the buffer pointer in BUFIN is incremented, ITASK is set to RTC6 and control

goes to RTC6.

If TEMP contains a line-feed character, a carriage return (4033_8) is stored in TEMP1 and control goes to RTC3A.

RTC3A checks the paper-tape mode flag in CS and if it is set, control goes to RTC4. If it is not set, the character saved in TEMP1 is stored in OUT and control goes to RTC4.

If TEMP contains none of the above-mentioned special characters, control goes to RTC4, where the value in TEMP is checked for a valid character ($40_8 \leq$ TEMP $\leq 137_8$). If the character is not valid, it is ignored and control goes to RTC6. If the character is valid, the value in TEMP is decremented by 40_8, left-shifted 17 bits, and saved in TEMP, and the low-order 17 bits of the A-register are cleared. TEMP now contains (teletype input character -40_8)/2 and the A-register has its sign bit on or off according to whether the teletype input character was odd or even. TEMP is used as an index into the TTY to display conversion table, TTYDIS. The left(right-adjusted) or right 6 bits of the table value are used according to whether the A-register is positive (even-valued teletype input character) or negative (odd-valued). The subroutine SDCH is called to store the 6-bit character in the PP buffer, and control goes to RTC6.

RTC6:

RTC6 is a dummy input task which immediately exits to TLOOPE.

WRCM:

WRCM is the system routine which is called only from the driver executive. See the partial description of WRCM contained in the Output Task Flow Narrative for TTY terminals. This description is concerned only with code beginning at WT12.

TEMP is set to 0 if ITASK=RTCH (EOD not received) or to 1 if ITASK=RTC6 (EOD received). If the break flag in CS is set, BK in the LSB is set to 1.

RD is checked in the LRB to determine where the input data should be written. If RD=0, the subsystem has requested that input on this line be terminated. Then if TEMP=0 (EOD not yet received), WRCM exits immediately without writing any data to CM. If TEMP≠0 (EOD received), control goes to WT50 (see description of WRCM in the Output Task Flow Narrative for TTYs). If RD=1, then the data goes into CIRCSTACK. The input buffer address is fetched from BUFIN and the location of the alternate buffer is computed. If the alternate buffer is not FREE then a call is made to subroutine CSTACK to write the alternate buffer to CIRCSTACK. If CSTACK detects that the EOD was written then control goes to WT50; else the FREE code is inserted into the FWA of the alternate buffer and WRCM exits.

If the alternate is FREE then TEMP is checked for the EOD input indication. When not EOD, WRCM exits. Else the current buffer is written to CIRCSTACK and control goes to WT50.

In the case where RD=3 {message buffer}, the low order digit in BUFIN is checked to see if at least 5 pp words of data {1 CM word} has been input. If not, TEMP is tested and WRCM exits if EOD has not been input. When the EOD input is flagged then control goes to WT35. Also, if at least 5 pp words are in, control goes to WT35.

At WT35, the FWA of the PP buffer is computed from BUFIN by clearing the low order digit and one CM word of data is written to the message buffer at the location contained in CBB. (CBB was set to RBB in ITTY when the line was initialized for input. Before any data is written, CBB is checked to insure that data is not written beyond the end of the line status table for this line). CBB is then incremented and the bytes 5, 6 and 7 are moved down to bytes 0, 1 and 2 in the 8 byte buffer. This is necessary since data may have been received into these bytes before WRCM could be called to write the first 5 bytes of data to CM.

If the EOD has not been received, the address in BUFIN is decremented by 5, the new CBB is written back to CM and WRCM exits. Otherwise control goes to WT50 because the EOD has now been written to CM.

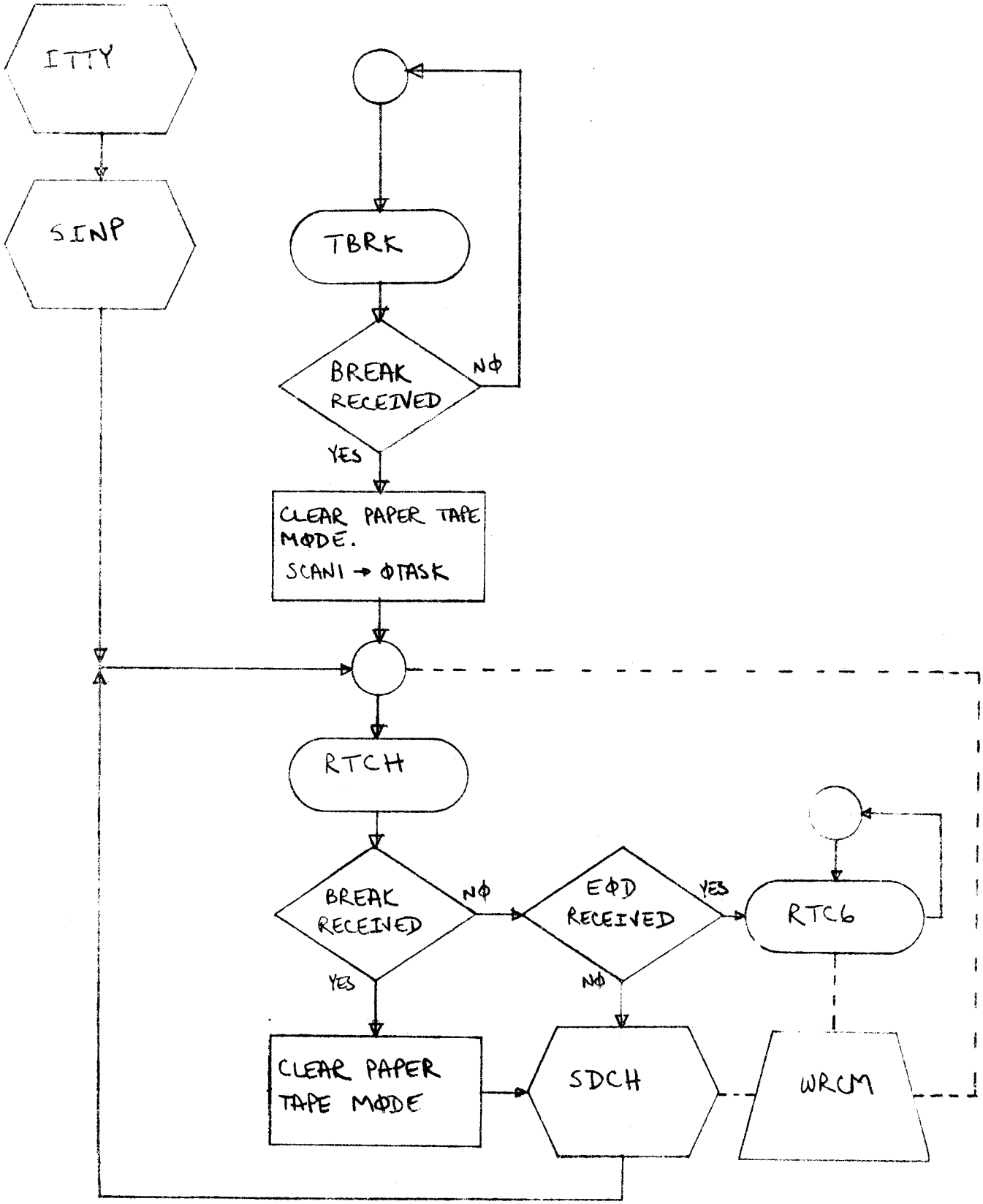
SDCH:

The subroutine SDCH is called by RTCH to store the display-coded teletype input into the PP buffer. The location in the PP buffer is obtained from BUFIN and the current position flag in CS is toggled. If the current position is now defined as the left, the 6-bit character is stored in the left half of the PP word and SDCH exits. If the current position is the right, the character is stored in the right half of the PP word. In this case, BUFIN must be incremented to the next PP word or, if the end of the 8 PP-word buffer is reached, BUFIN is set to the beginning of the buffer and SDCH exits.

8.1 Input Task Flow Diagram - Teletype Terminal

INPUT TASK FLOW DIAGRAM

TTY



9.0 MISCELLANEOUS ROUTINES

RXTFL:

RXTFL is a subroutine which performs the identical function of R.TFL. All references to CM are preceded by a call to RXTFL to verify the address and to add the current RA. If the address is within the FL, RXTFL exits with the absolute address in the A-register. Otherwise when a FL error occurs, the driver writes an error indication to the RESPOND word W.CPRES1 in the control point area. The second byte (from left to right) contains the location of the TFL fail and the two low order bytes contain IOD left adjusted in 6000 Display code. Control then goes to DROP.

DROP:

DROP is the exit routine for the driver. Control comes to DROP only from the subroutine RXTFL, which is a fatal error or from the main loop executive at EXT15, which is the normal exit procedure. DROP can only be called when MOVE=0 and the PP's output register is clear.

In order to drop, the PP resident program must be restored. The driver requests the loading of the PP program 1PR. 1PR sends the PP resident beginning at location 100_8 over the channel specified by the driver in 1PR's input register

{low order byte}. The driver and 1PR do a little handshaking with the channel in the following fashion:

1. The driver activates the channel at CHAN1 and keeps a monitor pause request M.PAUSE in its output register until 1PR disconnects the channel. {As soon as 1PR is loaded it disconnects the channel.}

2. The driver then activates the channel and does a 1000_8 word input starting at location 100_8. {In the mean time, 1PR hangs in a channel inactive loop until it is activated by the driver. Then 1PR does a 1000_8 word output from location 100_8.}

3. The driver disconnects the channel at CHAN5 and transfers control to location 1002_8. At 1002_8 the driver waits for its output register to clear, drops its channel(s), drops the PP and exits to the resident IDLE routine. (Note that the code beginning at 1002_8 came from 1PR).

RXDFM:

RXDFM is a subroutine which writes 1 CM word to the Day file line of the control point display area. The address of the message is in the A-register when RXDFM is called. The direct cell DFM contains the location in CM of the Dayfile line.

REQBUF:

REQBUF is the subroutine which obtains a free buffer from the POOL. The FWA of an available buffer is contained in the table JUNGL at the location contained in the direct cell JNEXT. If no buffer is available REQBUF exits with the A-register zero. Else JNEXT is incremented around the circular table JUNGL. The FWA of the buffer is stored in FROM and REQBUF exits with it also in the A-register.

GETBUF:

GETBUF is a higher level subroutine which calls REQBUF to obtain a POOL buffer but also initializes the table entries in BUFIN and BUFLM for 200 USER tasks. If no buffer is available then GETBUF sets RC=5 (no PP buffer available) and control goes to RCLC45 where a null buffer is queued via QCMW. If, however, a buffer is available, the FWA is placed in BUFIN and the LWA is placed in BUFLM. GETBUF then exits.

RETBUF:

RETBUF is the subroutine which returns a buffer to the POOL. If the A-register is non-zero when RETBUF is entered, then it is assumed to be the FWA of the buffer to be returned. Otherwise the FWA is computed from the LWA contained in BUFLM. The buffer is returned to the table JUNGL at the in location specified by the value in direct cell JEMPTY. JEMPTY is then

advanced in a circular fashion around JUNGL. The next slot in the JUNGL table is cleared to zero before RETBUF exits. If all the buffers are removed by REQBUF, then JNEXT would point to this zero entry, which would be a flag for the emptiness of JUNGL.

RELBUF:

RELBUF is a subroutine which is called from PETX and TIMERR to clean up any left over buffers. If ITASK=NEXT, then no buffers could be assigned to the line and RELBUF exits. Otherwise, the ETX was not expected or a timeout error occurred and a call is made to QCMW to queue a null buffer. This would eventually clear the "last buffer not written" flag. BUFIN is then tested for zero to determine if any buffer is still assigned. If BUFIN #0, then RETBUF is called with the A-register= zero prior to exiting.

GETCIR:

This subroutine is not assembled when RES=0, since it is used only for RESPOND. GETCIR writes one CM word to the current location in CIRCSTACK defined by the contents of direct cell CIL5. The CM word is setup in direct cells D.TW1-D.TW5. GETCIR computes the terminal number times 12 using LSWA and LFWA and places this in D.TW1. After writing this word to CM, GETCIR increments the CIRCSTACK entry number in CIL5 by one. If the end of CIRCSTACK is reached (when CIL5=CIRSTAX)

then CI15 is reset to zero. GETCIR always exits with the A-register non-zero. The location of the beginning of CIRCSTACK is contained in CIRP.

CSTACK:

This subroutine is not assembled when RES=0. CSTACK moves 4 pp words of data from the buffer specified by the FWA in FROM to the direct cells D.TW2-DTW5. If an odd number of characters were input, then a blank (55_8) is inserted in place of the zero character. GETCIR is called to write the CM word to CIRCSTACK. If an EOD was encountered, then CSTACK exits with the A-register zero.

JLINK:

This subroutine is not assembled when RES=0. JLINK takes the jungle unit number contained in direct cell JBB and fetches the next jungle unit of the chain as defined by RESPOND's link table. The FWA of the link table is contained in the instruction at JULP. The high order 10 bits in JBB are used as the relative CM address in the link table. This CM word is fetched and the low order 2 bits of JBB are used as the byte number. This byte is then stored into JBB as the new jungle unit number. JLINK exits with JBB in the A-register.

RJUNP:

This subroutine is not assembled when RES=0. RJUNP converts the jungle unit number contained in JBB to a relative CM address within RESPOND's field length. JBB is multiplied by 8 and added to the FWA of the jungle unit area contained in the instruction JUNP.

10.0 SUBSYSTEM ZERO - TRANSIENT PP's, 1SD AND 1ID

10.1 Subsystem Zero Narrative

PROGRAM - 1SD
ROUTINE - TIOD

TIOD is a transient program in monitor recall that performs the auxiliary tasks required by IOD. These tasks include 1) loading subsystems, 2) assigning the supervisory console and central display among the subsystems, 3) activating the storage allocator, 4) requesting field length changes for the allocator, and 5) creating the second line of the control point dayfile message. After initializing the PP, TIOD begins by scanning the entire line status table and creating two internal tables with one entry for each subsystem. The first table contains the count of the line assigned to each subsystem and is used to create the second line of the control point dayfile message. The second table contains flags that are set if any line is ready to be serviced by the subsystem to which it is assigned. In addition to creating the tables, TIOD processes lines which are assigned to subsystem zero but are not connected (i.e., have their data set hung up). TIOD passes these lines back to subsystem zero when they reconnect. After the line status table scan, TIOD procedes to perform an initial scan of the subsystem table. The purpose of this scan is to determine if the supervisory console and/or the central display are currently assigned, and to determine if any

subsystem requires the storage allocator. Following the initial scan, TIOD performs the final subsystem table scan. During this scan the second line of the control point dayfile message is generated, and assignment of the supervisory console and/or the central display is performed. For those subsystems which are not actively running but require attention, TIOD will load their main PP program and/or update the subsystem status. After the second subsystem table scan is complete and if the allocator is in recall, TIOD will request a field length change to the last value specified by the allocator, and then activate the allocator if it is required by some subsystem. Finally TIOD will write the second line of the control point dayfile area to CM, request a 1/16 monitor recall of itself, and then overlay itself with the last subsystem PP program to be loaded, or drop the PP.

The input register of a PP loaded by ISD will be of the form:

PP program	Name	CP no.	SSB		SYSPTR

(See Section 3.0 for description of SYSPTR and Section 6.1 for description of SSB).

PROGRAM-LSD
ROUTINE - CTFL

CTFL is a subroutine that calls RTFL to check CM addresses, relative to RA, and then add RA to the given address. If the address is outside the field length, the system failure procedure is followed {Section 11.1} and the PP hangs until an operator drop is issued.

PROGRAM - LID
ROUTINE - SIDC

SIDC is the primary routine of subsystem zero and provides the link between IOD and IDLE, the higher level routine of subsystem zero. SIDC acts as a subroutine which is called by IDLE to perform I/O on a single line. The primary duties of SIDC are to 1) cycle through all the lines in the line status table that are assigned to subsystem zero, 2) request retries on error conditions, and 3) complete processing of an operator drop request. SIDC begins by initializing the PP memory and checking the operator drop flag. SIDC then enters its main loop which scans the entire line status table, and processes those lines which are assigned to subsystem zero and are connected (i.e., are not being processed by LSD). If the operator drop flag is not set, processing the line consists of 1) checking to see if the line has disconnected and if so returning the line to LSD for processing, 2) checking to see if an I/O error has occurred and if so retrying the transmission, and 3) returning to the last call mode by IDLE providing that neither 1) or 2) occurred. If the operator drop flag is set, processing consists of checking an internal drop flag (unique to each line) that determines at what stage of dropping the line is. This generally requires three passes (reloads) to 1) force completion of any I/O in

progress, 2) reinitialize IDLE to issue the operator drop message, and 3) wait until the operator drop message is complete. Any I/O errors are ignored. After a return is made to IDLE, and SIDC is called again, SIDC will then cycle to the next line table entry and repeat the above procedure. When the scan is complete, SIDC will set W.CPRESL for IOD if an operator drop was requested and all the lines have completed dropping. Finally SIDC resets the subsystem status and drops the PP.

PROGRAM - LID
ROUTINE - IDLE

IDLE is the subsystem zero control routine that calls SIDC to process a single line. All calls to IOD are initiated by IDLE and not by SIDC. After being initially entered, IDLE calls SIDC to perform the final request made by the last subsystem assigned to the line, and then checks for an operator drop. If a drop is in progress, IDLE calls SIDC to issue the operator drop message, and signal that the drop is complete. Otherwise IDLE examines the new subsystem name provided by the last subsystem, or calls SIDC to issue the terminal idle message and poll for a new subsystem name. When a new subsystem name is available, IDLE scans the subsystem table and compares the names of available subsystems with the name entered. If a match is found, the line is assigned to the new subsystem and SIDC is called to look at the next line. Otherwise IDLE calls SIDC to issue the subsystem not available message, and branches back to the point where terminal idle is issued until the next subsystem name is entered.

PROGRAM - LID
ROUTINE - CTFL

CTFL is a subroutine that calls R,TFL to check CM addresses, relative to RA, and then add RA to the given address. If the address is outside the field length, the system failure procedure is followed {Section 11.1} and the PP hangs until an operator drop is issued.

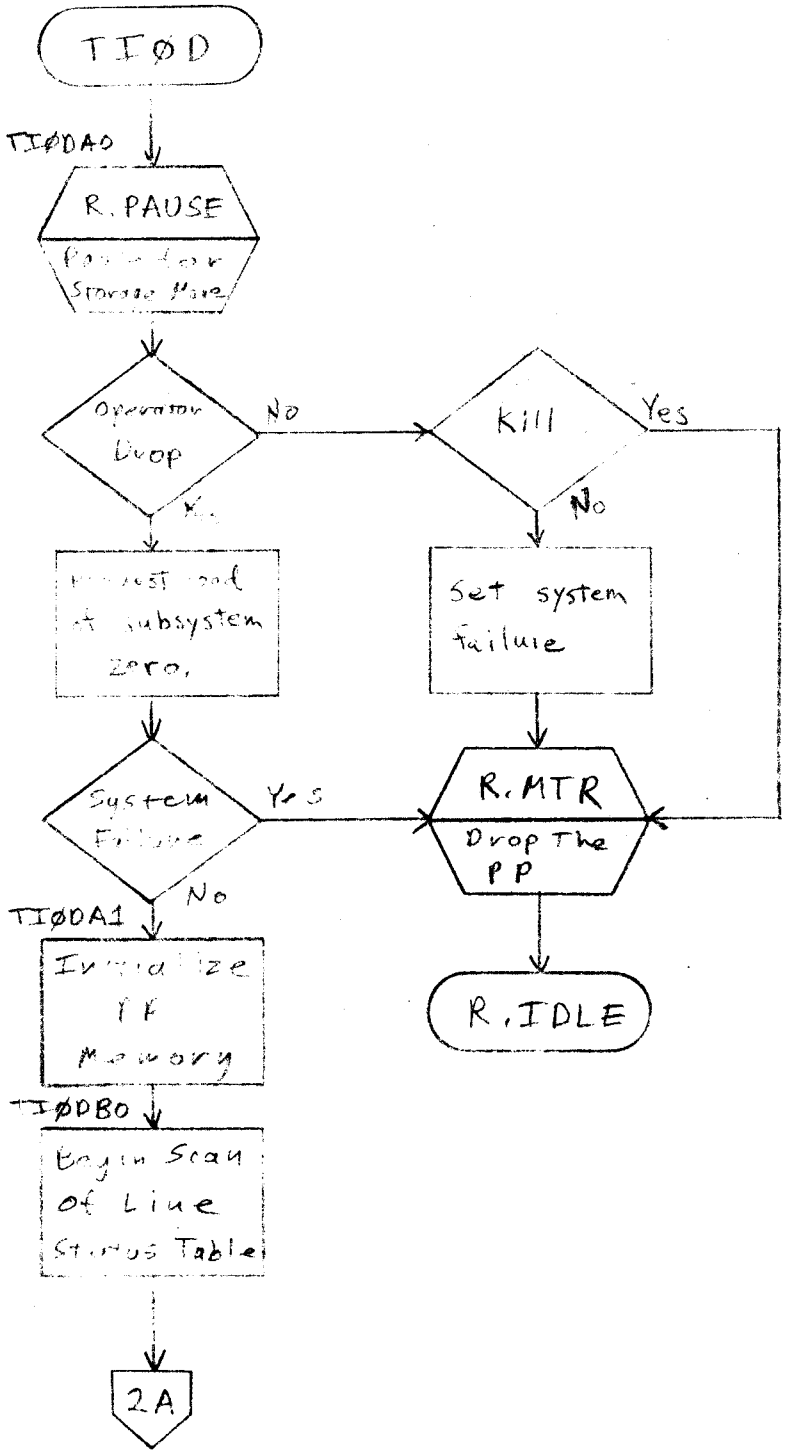
PROGRAM - LID
ROUTINE - ENDC

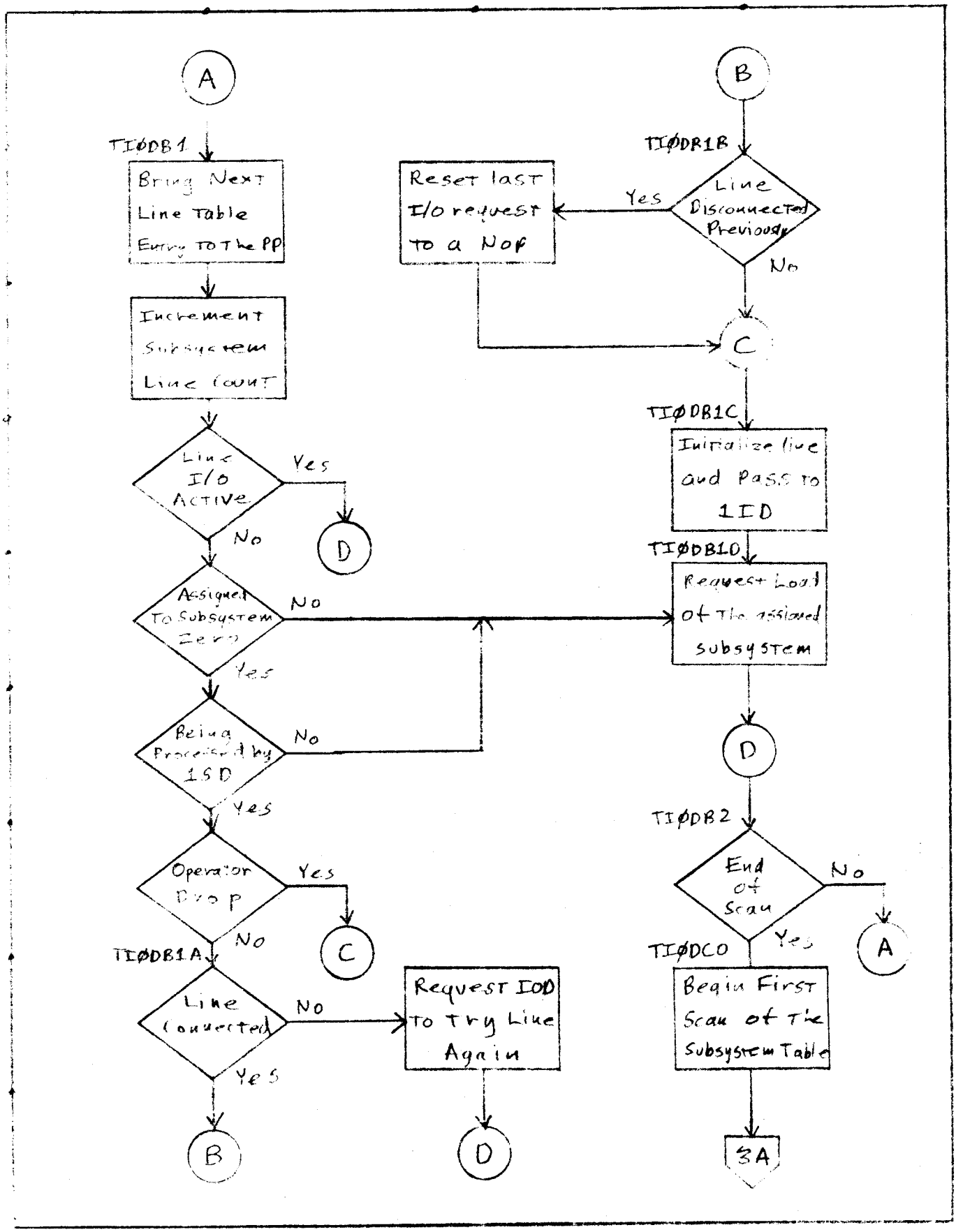
ENDC is a subroutine to insert a CR, EOM or EOM at the end of a message depending on whether the line is a TTY or CRT.

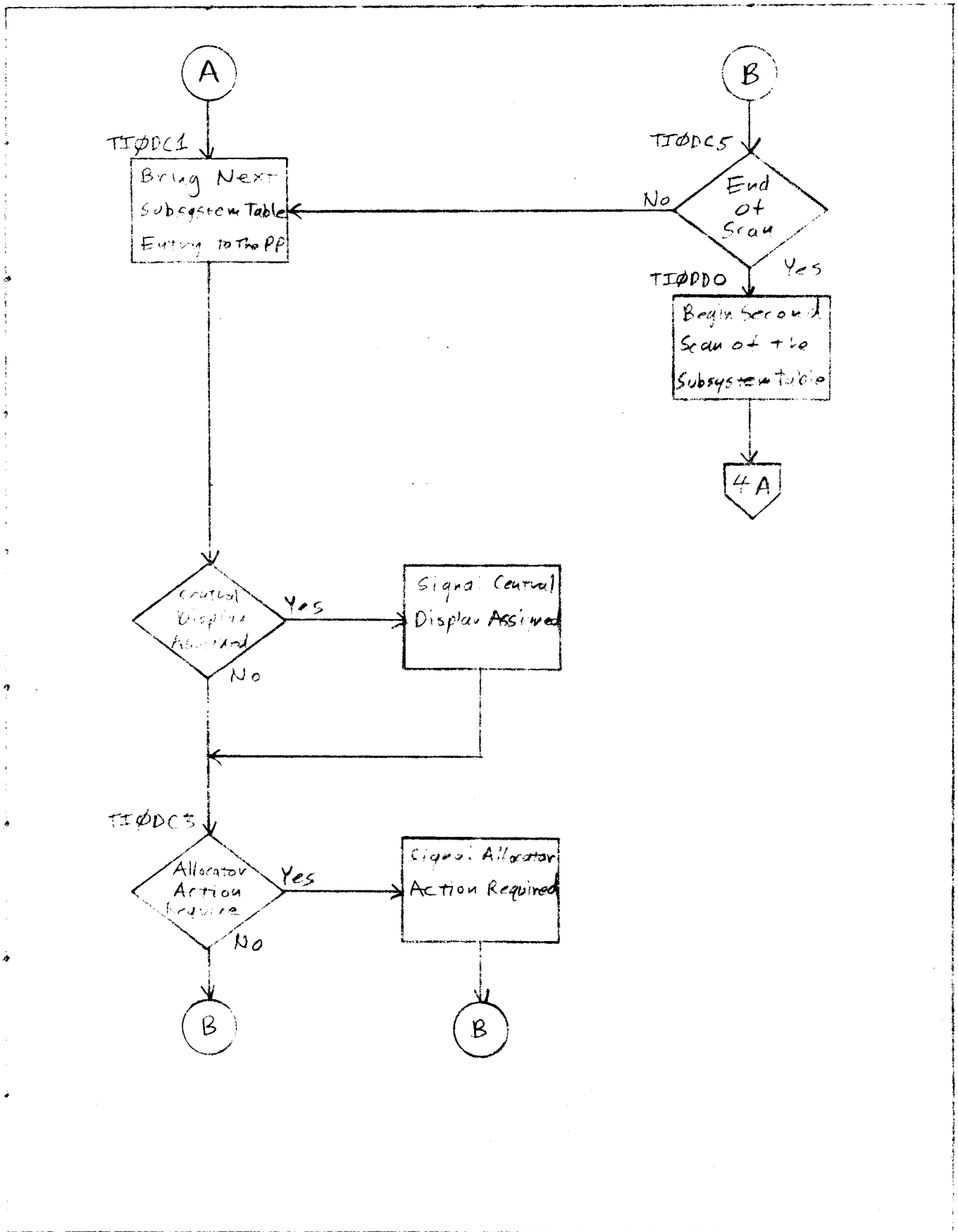
10.2 Flow Charts for 1SD and 1ID

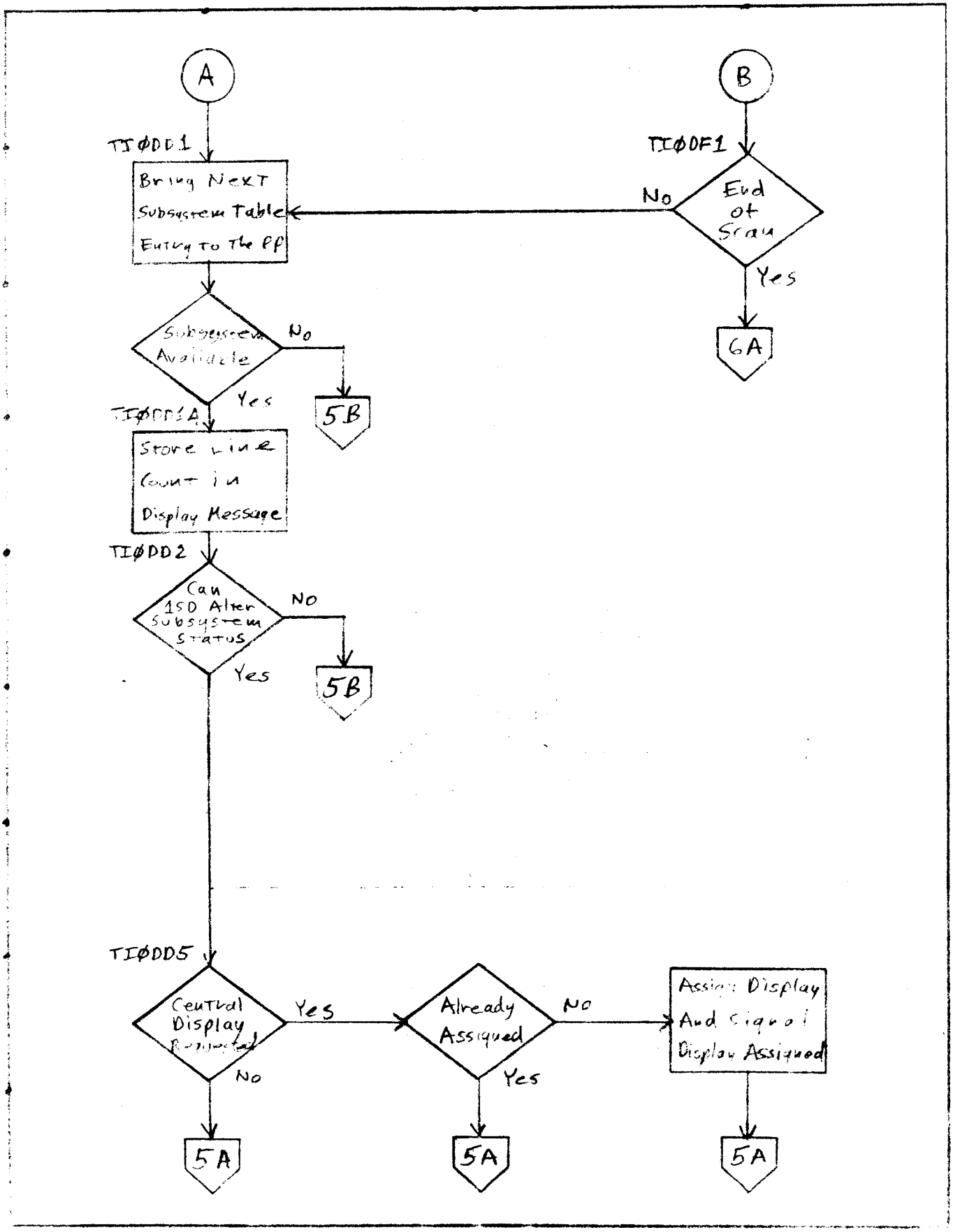
1SD

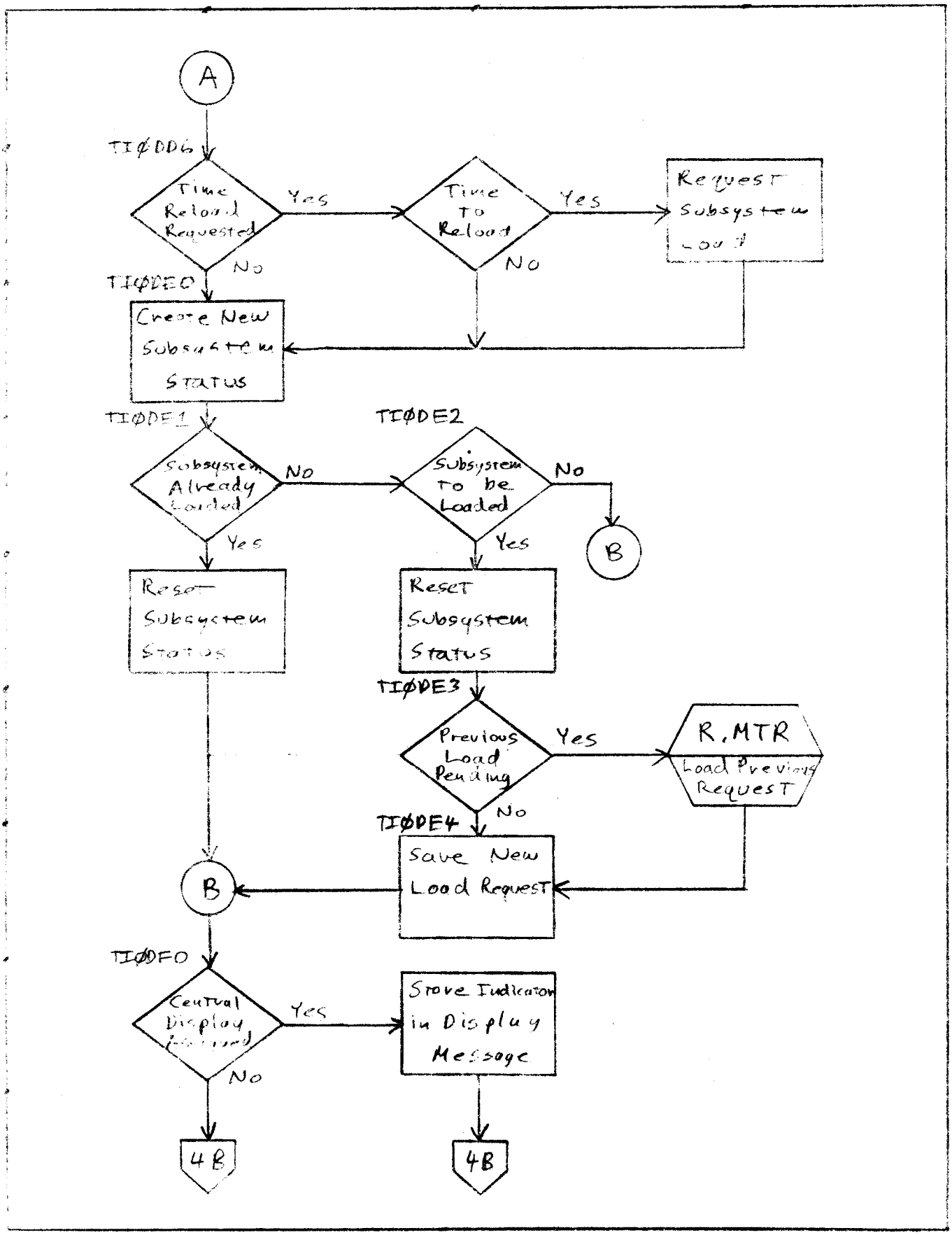
Page 1

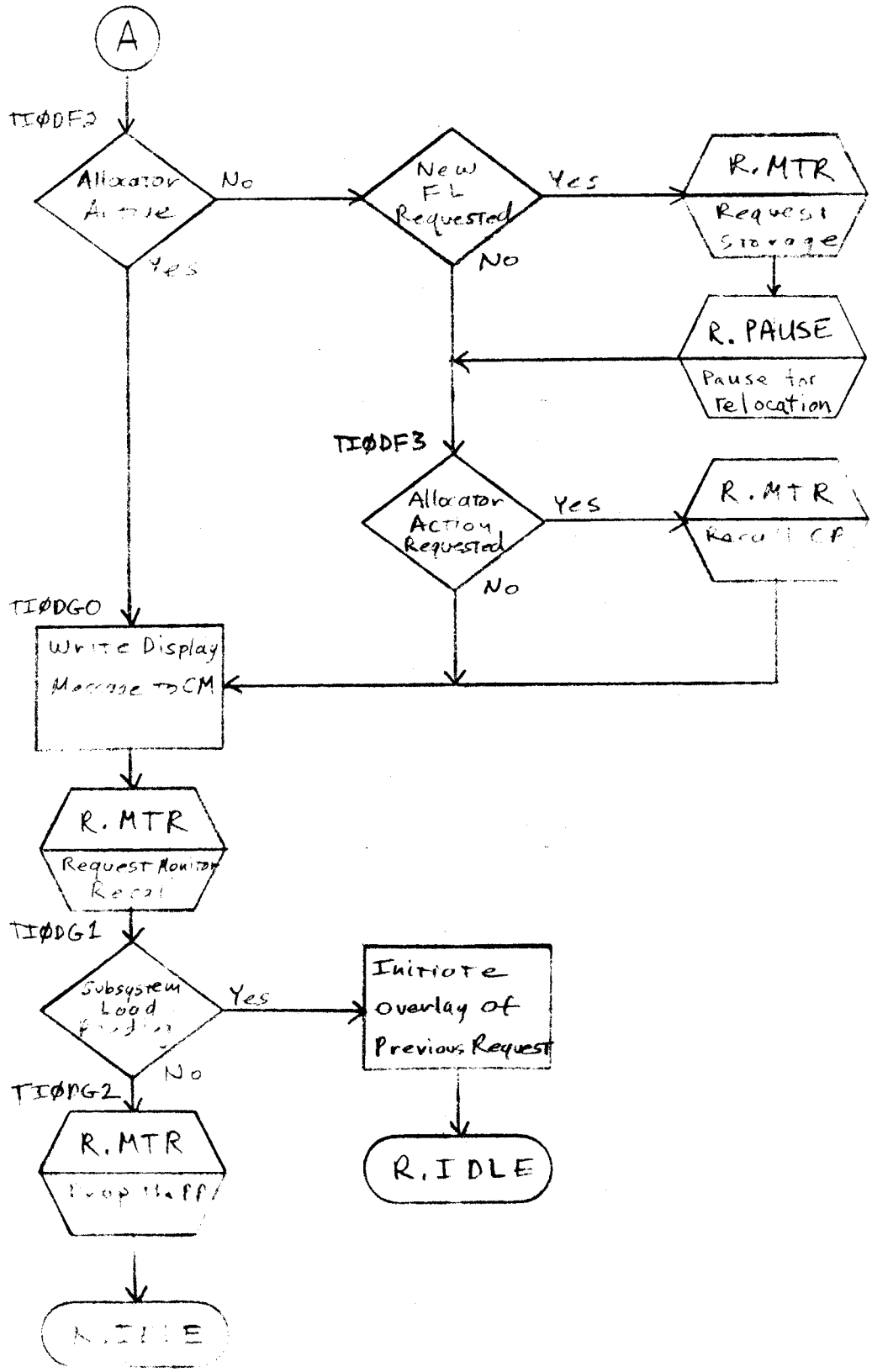

TIØD
TIØDA0
R.PAUSE
Operator Drop
No
Yes
Kill
No
Set system failure
System Failure
Yes
No
R.MTR
Drop The PP
TIØDA1
Initialize PP Memory
TIØDB0
Begin Scan of Line Status Table
2A
R.IDLE


A
TIØDB1
Bring Next Line Table Entry TO The PP
Increment Subsystem Line Count
Line I/O Active
Yes
No
D
Assigned To Subsystem Zero
No
Yes
Being Processed by 1SD
No
Yes
Operator Drop
Yes
No
C
TIØDB1A
Line Connected
No
Yes
B
B
TIØDR1B
Line Disconnected Previously
Yes
Reset last I/O request to a Nop
No
C
TIØDB1C
Initialize line and Pass to 1ID
TIØDB1D
Request Load of The assigned subsystem
D
TIØDB2
End of Scan
No
Yes
A
TIØDC0
Begin First Scan of the Subsystem Table
3A
Request IOD to Try Line Again
D

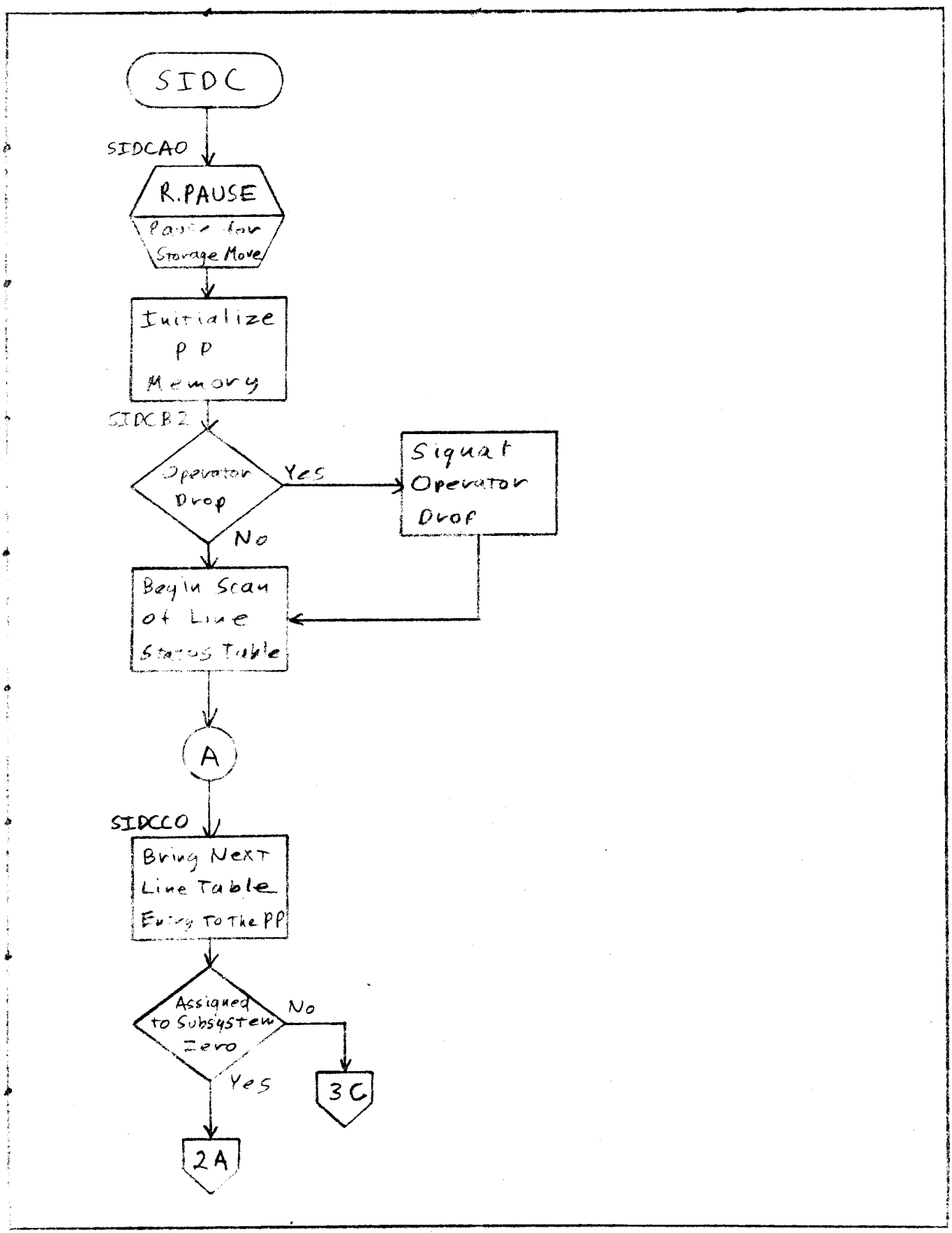

A
B
TIØDC1
Bring Next Subsystem Table Entry to The PP
TIØDC5
End of Scan
No
Yes
TIØDDO
Begin Second Scan of the Subsystem Table
4 A
Central Display Assigned
Yes
No
Signal Central Display Assigned
TIØDC3
Allocator Action Require
Yes
No
Signal Allocator Action Required
B
B

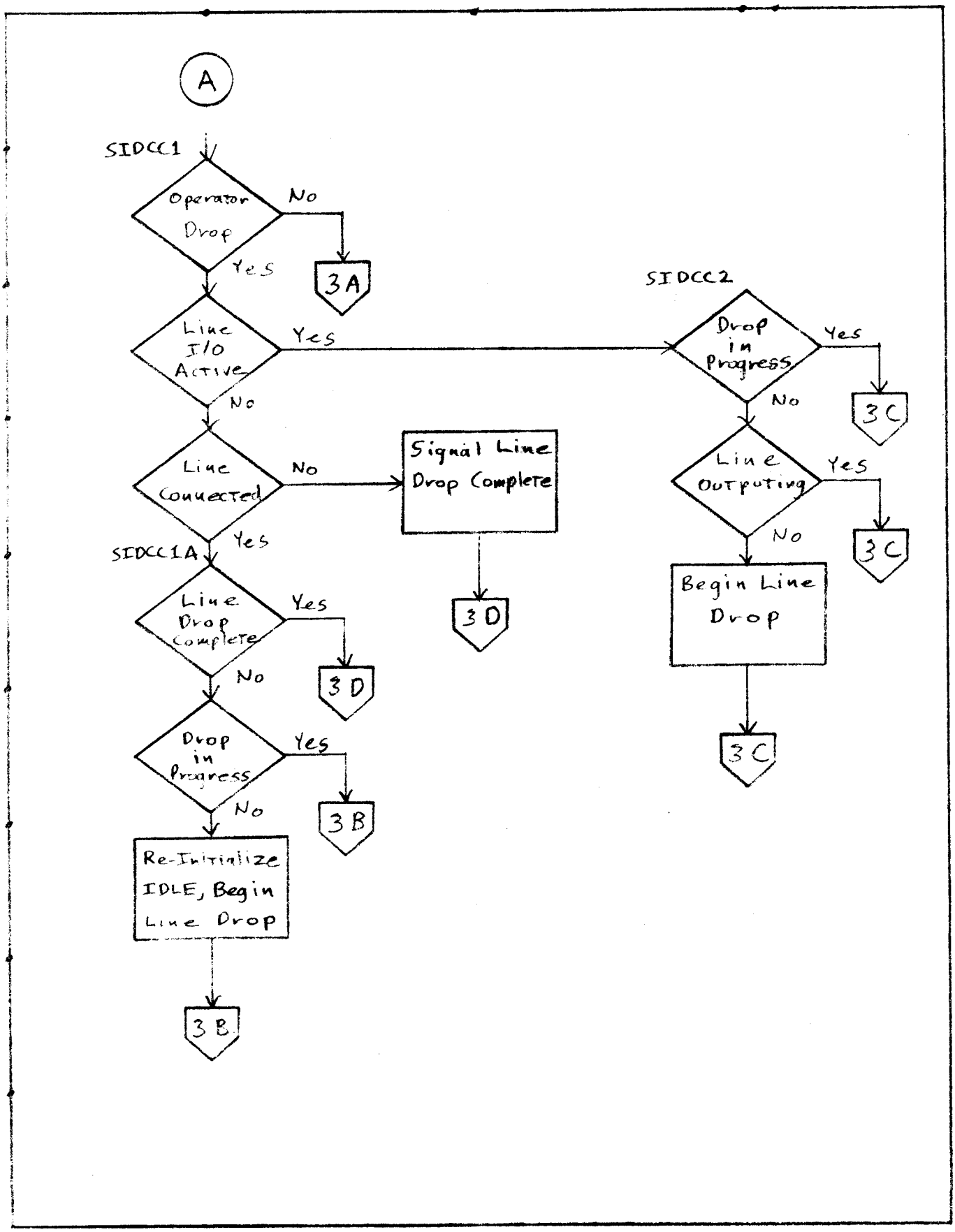

A
B
TIØDD1
TIØDF1
Bring Next Subsystem Table Entry to the PP
No
End of Scan
Yes
6A
Subsystem Available
No
Yes
5B
TIØDD1A
Store Line Count in Display Message
TIØDD2
Can 1SD Alter Subsystem Status
No
Yes
5B
TIØDD5
Central Display
Yes
No
Already Assigned
No
Yes
Assign Display And Signal Display Assigned
5A
5A
5A

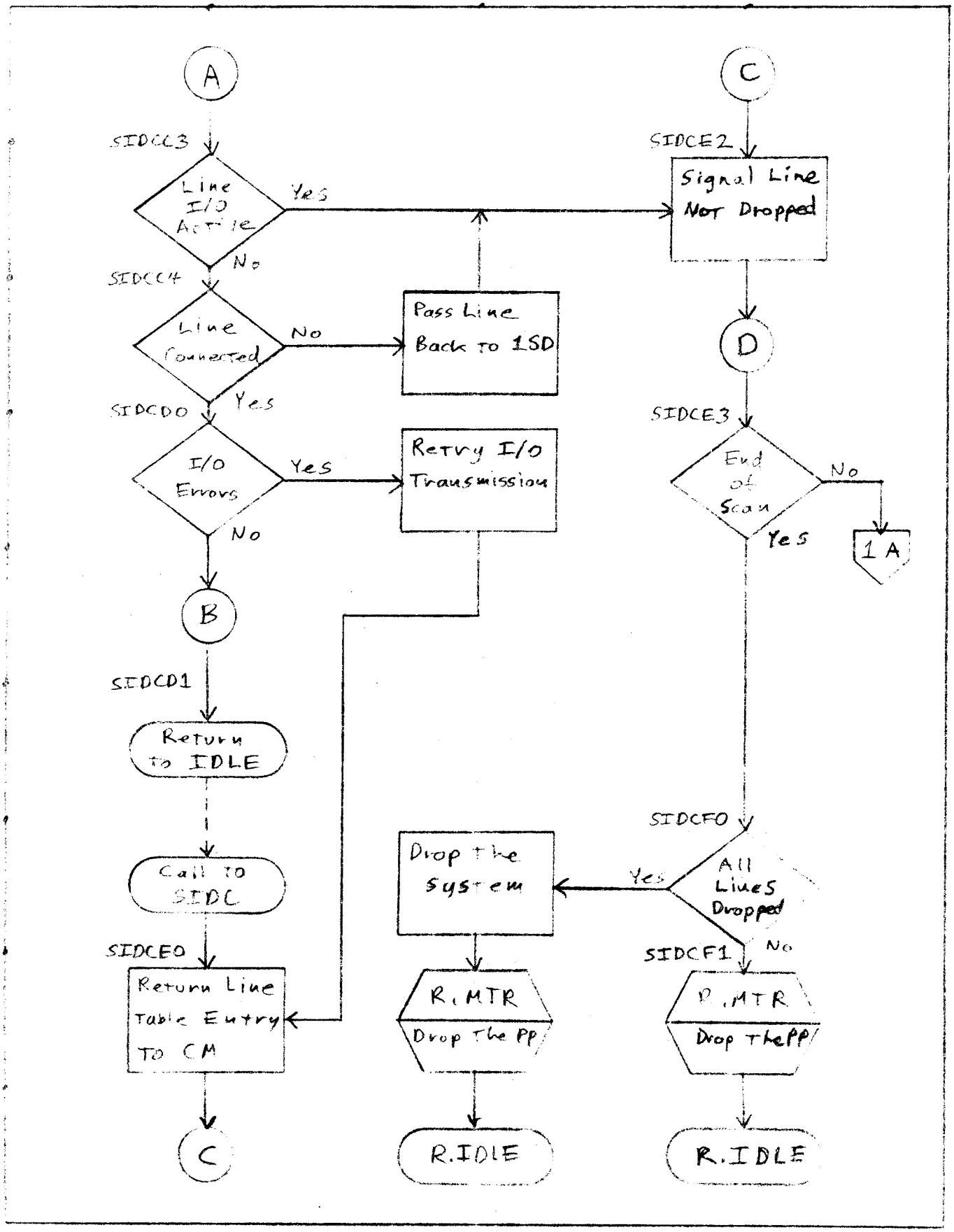

A
TIØDD6
Time Reload Requested
Yes
Time To Reload
Yes
Request Subsystem Load
No
No
TIØDE0
Create New Subsystem STATUS
TIØDE1
Subsystem Already Loaded
No
TIØDE2
Subsystem to be Loaded
No
B
Yes
Yes
Reset Subsystem Status
Reset Subsystem Status
TIØDE3
Previous Load Pending
Yes
R.MTR
Load Previous Request
No
TIØDE4
Save New Load Request
B
TIØDF0
Central Display
Yes
Store Indicator in Display Message
No
4B
4B

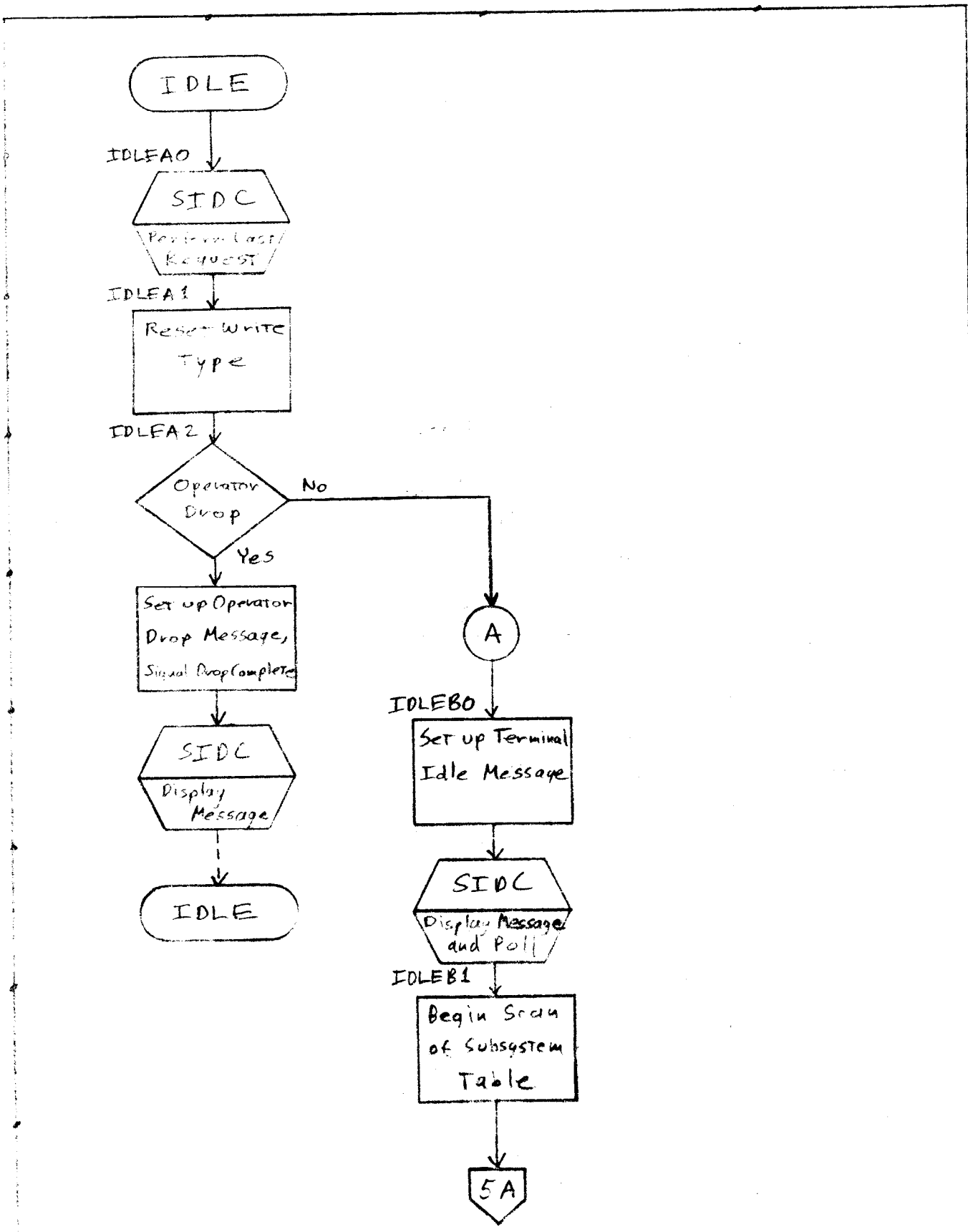

A
TIØDF2
Allocator Active
No
Yes
New FL Requested
Yes
No
R.MTR
Request Storage
R.PAUSE
Pause for relocation
TIØDF3
Allocator ACTION Requested
Yes
No
R.MTR
TIØDGO
Write Display Message to CM
R.MTR
Request Monitor Recal
TIØDG1
Subsystem Load Pending
Yes
No
Initiate Overlay of Previous Request
TIØDG2
R.MTR
R.IDLE
R.IDLE

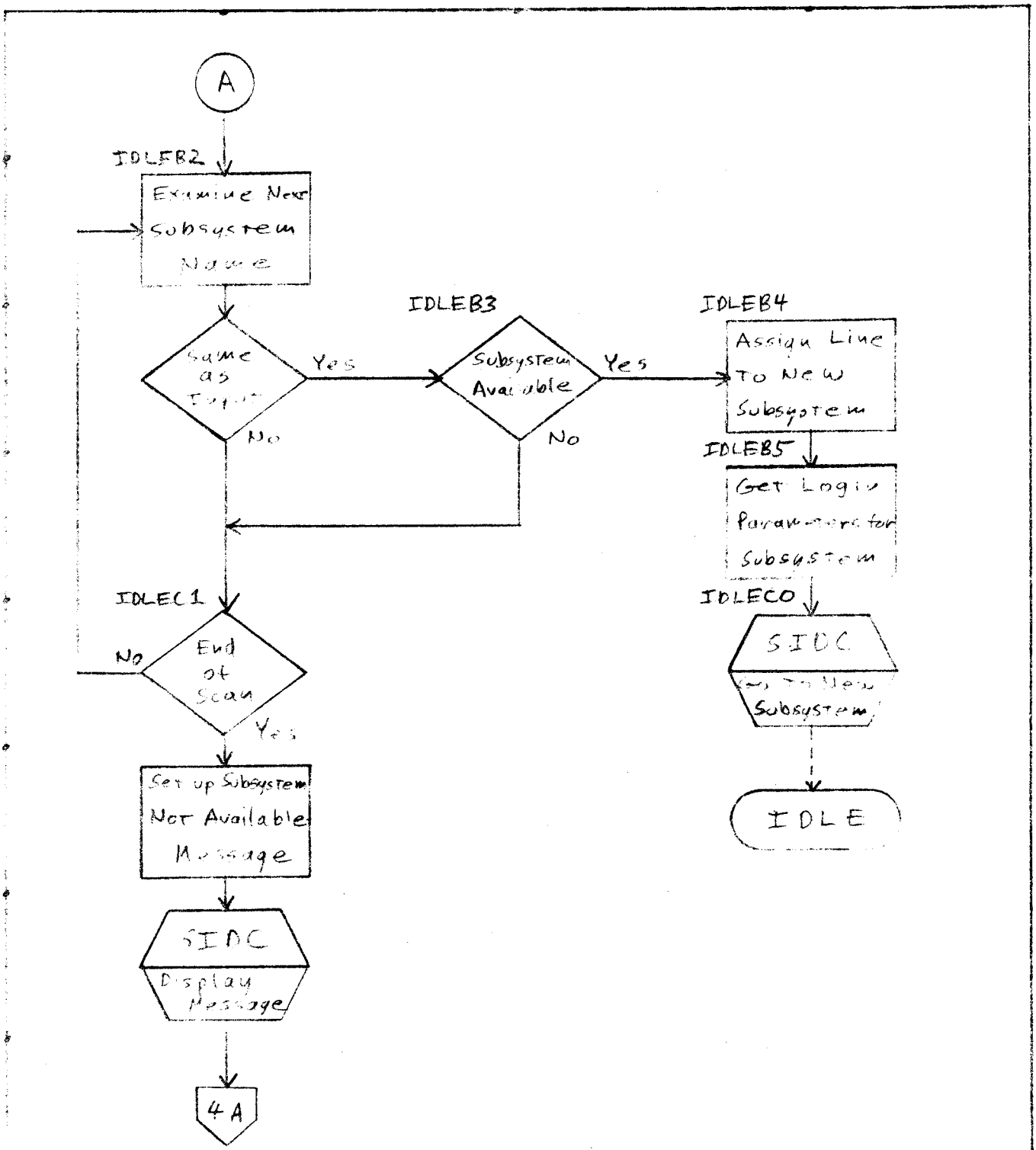
1ID


SIDC
SIDCA0
R.PAUSE
Pause for Storage Move
Initialize P P Memory
SIDCB2
Operator Drop
Yes
Signal Operator Drop
No
Begin Scan of Line Status Table
A
SIDCC0
Bring Next Line Table Entry To the PP
Assigned to Subsystem Zero
No
Yes
3C
2A


A
SIDCC1
Operator Drop
No
Yes
3A
Line I/O Active
Yes
No
Line Connected
No
Signal Line Drop Complete
3D
Yes
SIDCC1A
Line Drop Complete
Yes
3D
No
Drop in Progress
Yes
3B
No
Re-Initialize IDLE, Begin Line Drop
3B
SIDCC2
Drop in Progress
Yes
3C
No
Line Outputing
Yes
3C
No
Begin Line Drop
3C


A
SIDCL3
Line I/O Active
Yes
No
SIDCL4
Line Connected
No
Yes
Pass Line Back to 1SD
SIDCD0
I/O Errors
Yes
No
Retry I/O Transmission
B
SIDCD1
Return to IDLE
Call to SIDC
SIDCE0
Return Line Table Entry To CM
C
C
SIDCE2
Signal Line Not Dropped
D
SIDCE3
End of Scan
No
Yes
1A
SIDCF0
All Lines Dropped
Yes
No
Drop the System
SIDCF1
R.MTR
Drop The PP
R.MTR
Drop ThePP
R.IDLE
R.IDLE


IDLE
IDLEA0
SIDC
Retrieve Last Request
IDLEA1
Reset Write Type
IDLEA2
Operator Drop
No
Yes
Set up Operator Drop Message, Signal Drop Complete
SIDC
Display Message
IDLE
A
IDLEB0
Set up Terminal Idle Message
SIDC
Display Message and Poll
IDLEB1
Begin Scan of Subsystem Table
5A


A
IDLEB2
Examine Next Subsystem Name
Same as Input
Yes
No
IDLEB3
Subsystem Available
Yes
No
IDLEB4
Assign Line To New Subsystem
IDLEB5
Get Login Parameters for Subsystem
IDLEC0
SIDC
Subsystem
IDLE
IDLEC1
End of Scan
No
Yes
Set up Subsystem Not Available Message
SIDC
Display Message
4A

11.0 TERMINATION ROUTINE NARRATIVE (1PR)

PROGRAM-1PR
ROUTINE-PPR

PPR initializes the PP and stores the channel number in the I/O instructions of CHAN.

PROGRAM - LPR
ROUTINE - DROP

DROP performs the final section of the system failure procedure {Section 11.1 } and then waits until an operator DROP or KILL request.

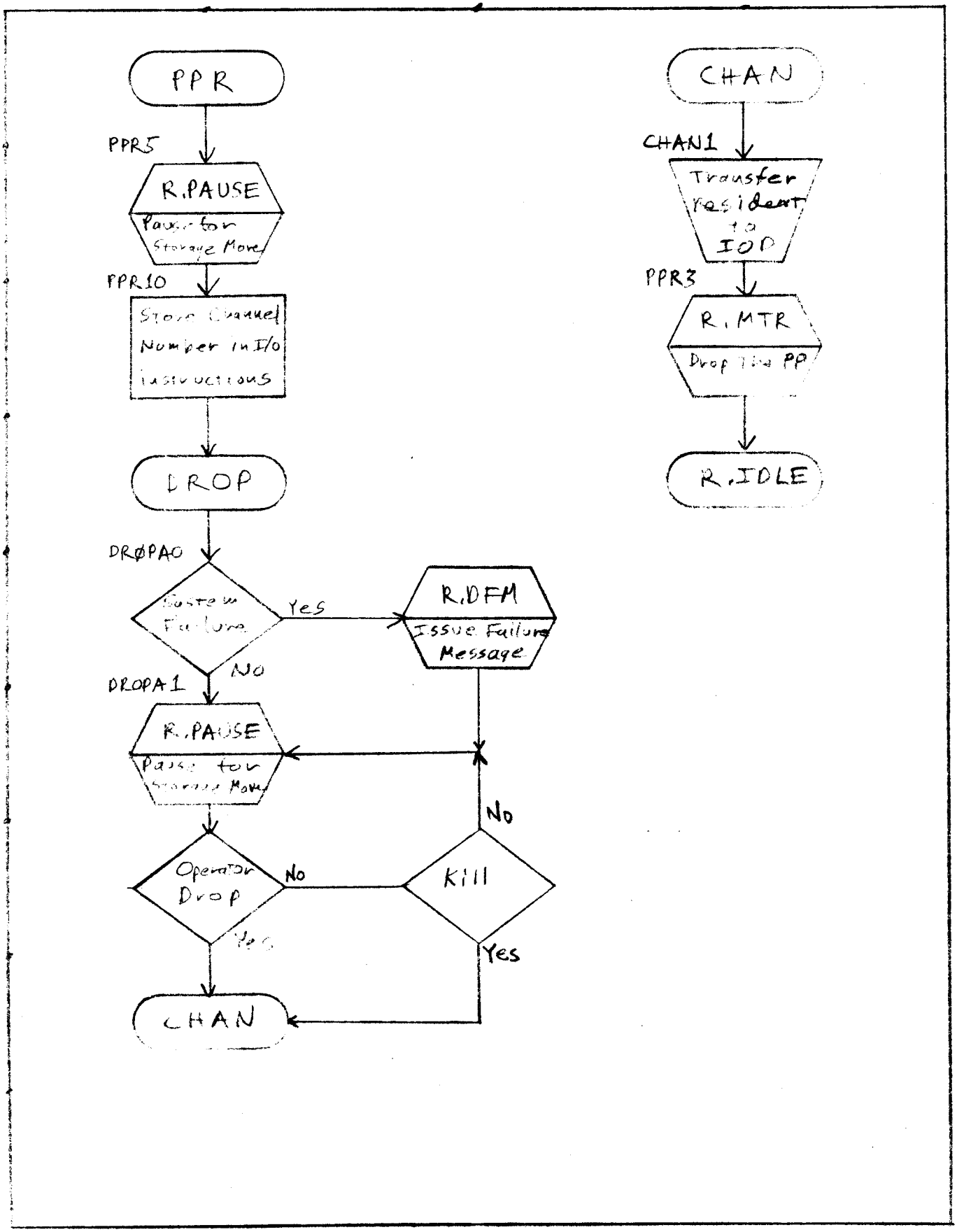
PROGRAM - 1PR
ROUTINE - CHAN

CHAN transfers a new copy of the PP resident to IOD via the specified I/O channel, and then drops the PP.

11.1 Termination Routine Flow Chart

1PR

Page 1


PPR
PPR5
R.PAUSE
Pause for Storage Move
PPR10
Store Channel Number in I/O Instructions
DROP
DRØPA0
System Failure
Yes
R.DFM
Issue Failure Message
No
DROPA1
R.PAUSE
Pause for Storage Move
Operator Drop
No
Kill
No
Yes
Yes
CHAN
CHAN
CHAN1
Transfer resident to IOP
PPR3
R.MTR
Drop This PP
R.IDLE

12.0 TABLE USAGE

12.1 One for Each Port

LS: (200 USER Terminals only)

LS is initialized by SCAN when an output is ready to begin (i.e., just before SSYNC is set into OTASK). LS contains:

CC,SB,0,SA(4),COUNT(5)

The 5 bit field COUNT is used by SSYNC to determine how many SYNC codes must be sent out prior to sending the SOM code. SCAN initially places the value 4 into COUNT. The field SA contains the low order 4 bits of the site address of the 200 USER Terminal which is sent out after the SOM code. SB is the current sequence bit which is placed in the 2^4 position of the station address character if the message is a WRITE message ($3 \leq MT \leq 6$). For a POLL or ALERT message the 2^4 bit is cleared. CC is the POLL indicator bit which is =1 if this message is a POLL. The station address and control code characters are selected based upon CC.

After the control code character has been output, LS contains the SYNC word count to be sent after a RESET, CLEAR or DIAGNOSTIC WRITE code before the data. This is initialized to 7 by the SCNTL task. When the SYNCs have been sent, LS is set to zero as a first time flag for READCM, to be used in case any data is sent to the terminal.

When the ETX is sent to the terminal, LS is cleared to zero. During the receive, the RSB byte is assembled in LS. In RMPC, LS is placed into RSB.

No data on this page

WS: (200 USER Terminal only)

WS is initialized by SCAN when an output is ready to begin. WS contains LRB, except for RESPOND (ID=NRES) where CR is set equal to the DR bit from the previous RSB.

MT is used as an index into the MESS table at SCNL to fetch the output message control code. DT is used at SNDT as an index into the DT table to fetch the proper E code for the 200 USER Terminal.

CR is used by SCNTL to determine if a carriage return should be sent before the data for a normal write message (MT=3).

During receiving at RESYNC, WS is cleared except for RD. At the end of receiving at RMPC, WS will contain:

QCR,NI,EX,RD(2),QC(2),BN(5)

QCR is set by WRITCM after writing the CIRSTACK entry containing an EOD. (See QC=3) QCR is used as a flag to RMPL. When QCR is set, RMPL clears QCR and requests QCMW to queue request 1 for WRITCM (QC=1). RMPL then waits for WRITCM to clear BN.

NI is set to one by WRITCM when initializing a buffer for writing to CIRCSTACK. When the complete buffer has been written, WRITCM clears NI.

EX is set by PETX when an ETX is received.

RD is used by WRITCM to determine the type of CM buffer to receive the data.

QC is the "queue-last-buffer" flag for PETX. When the device status (E1, E2, or E3) is received, this flag is set. QC is also set at RMPL for RESPOND buffers. QC values have the following meaning to WRITCM:

1 - write U002 followed by RSB into CIRCSTACK

2 - write a partial buffer (last buffer)

3 - write a null buffer. Also if the write is to CIRCSTACK set QCR in WS.

CS:

The high order bit of CS determines the type of terminal on this line. If the bit is set (CS=4xxx), the line is a Teletype. If clear, the line is a 200 USER, and only the 3 low order bits of CS are used. CS contains the value of CS in LSB and is gradually incremented from CS=0 in SCAN up to CS=6 in RMPC. CS is incremented in the following tasks:

VALUE	TASK	MEANING
0	SCAN	line inactive
1	SC60	waiting to output SYNCs
2	SSYNC	waiting to output SOM
3	SSOM	outputting through MPC
4	RESYNC	waiting to input
5	RSOM	inputting through ETX
6	RMPC1	input complete (MPC)

After the last input buffer is written to CM by WRITCM, WS is no longer needed. This condition is detected at RMP1, which stores CS in WS and sets CS=0. Whenever CS=0, LOOP will immediately go to the current output task (OTASK) without regard for the multiplexor input word status. At RM 10, CS (now in WS) is inserted into the CS field of LSB. If a TIMEOUT occurs with CS=4 and ID=0 and PLB=0, then the site address is incremented at RMP15.

When the line is a TTY, CS contains:

TY,0,0,0,0,0,PT,0,EF,0,BK,CP, where TY=1

PT is the paper tape input flag which is set at SC43 when DT=3. If PT is set, sending of a semi-colon or carriage return at RTC3A is crippled. PT is cleared when a break character is received at TBRK or RTC1. The error flag, EF, is set at WT34 when an illegal message buffer address is detected. It is transferred from CS to the RSB at

WT71 before TL is set. The break flag, BK, is set at TBRK when a break character is input when the TTY is in the output mode. The TTY is put in the input mode and all output is stopped. When WRCM is executed for this line, BK is transferred from CS to the LSB. CP indicates the position in the current PP word of the last character received. It is initialized to 0 (right character) at ITTY and toggled whenever SDCH is called to store a character.

BUFIN/BUFOUT:

These two tables must occupy the same locations. BUFIN is used only while the driver is inputting from the terminal, while BUFOUT is used only during the output phase.

For 200 USER Terminals, BUFIN/BUFOUT is zero when no data is being processed to or from a POOL buffer. When non-zero, it contains the current PP location of the data character being processed. Before the data portion in a READ message is actually received, BUFIN will always be zero. After the READ control code has been processed (at RCNTL-RCN1) and the next character is not an escape, a buffer is requested at RCLB via GETBUF, which sets both BUFIN and BUFLM. As the first of each pair of characters (after the first pair) is received at RCLD1, BUFIN is incremented. When the second of each pair of characters is processed at RCRA4, the current value of BUFIN is compared with the last location of the PP POOL buffer contained in BUFLM. When the contents of BUFIN and BUFLM are equal, BUFIN is cleared and a call to subroutine QCMW is made to queue a request to write the buffer of data to CM. Note that the buffer itself is not returned to the POOL until WRITCM has completed processing of the data. If less than 80 characters are received because an E code is detected beginning at RCLC25, then a partial buffer is queued at RCLC40 and BUFIN is cleared. When a Timeout error occurs during input (CS$\geq$4), the call to RELBUF at TIMERR will result in a call to RETBUF if BUFIN is non-zero. This will return the POOL buffer and clear BUFIN. RELBUF is also called from PETX in case an

ETX was received before an E code and a buffer is still assigned.

BUFOUT is set by READCM with the PP address of a POOL buffer containing data to output. When the first character of each pair in a PP word is needed for output, BUFOUT is incremented by one (SLC). At SRC the second character is output and the contents of BUFOUT and BUFLM are compared at SRC2. If they are equal, control goes to EJUNG and RETBUF returns the buffer to the POOL and BUFOUT is cleared. When the output buffer is only partially filled, the EOB or EOD code is detected at SLC20 and RETBUF is called. If a Timeout error occurs during output (CS$\leq$3) and BUFOUT is non-zero, then RETBUF is called at TIM1. BUFIN/BUFOUT must always be zero when an output begins to the terminal; otherwise, a timeout error would result in a buffer being incorrectly returned to the POOL.

For TTY lines, BUFIN is always non-zero and contains the current PP word for input into the permanent input buffer. During initialization at IN402, each TTY line is assigned a fixed 8 PP word buffer beginning on an 8 PP word boundary. Only the low order octal digit of BUFIN varies from 0 to 7. While inputting to CIRCSTACK for RESPOND, the buffer uses two alternating 4 PP word (8 character) buffers. While one 8 character part is filling, WRCM has at least 8 character times (800 milliseconds) to write the other part to CM. When inputting to the message buffer, an entire CM word is assembled before a write to

CM is accomplished. Therefore, whenever the low order octal digit is 5 or greater, WRCM will write to CM (600 milliseconds). BUFOUT is not used by TTY lines.

Note that when TTYs are used, the POOL buffers must begin on an 8 PP word boundary, since they are cracked into five 8-PP word buffers for TTY's.

BUFLM/TOUT:

These two tables occupy the same locations to save space in the PP. BUFLM is used only for 200 USER Terminals, and TOUT is used only while outputting to TTYs.

For 200 USER Terminals, BUFLM contains the LWA of the POOL buffer currently being used for either output or input. For output, READCM sets BUFLM when a POOL buffer is obtained. When a buffer is needed during input, GETBUF sets BUFLM. The length of a POOL buffer is LJB. RETBUF and QC10 use BUFLM to compute the FWA of the buffer being returned or queued. The description of BUFIN/BUFOUT mentions the end of buffer tests made using BUFLM at RCRA and SRC2. Also at RCLC3, when the EOD is to be inserted into a buffer after the data, BUFLM is used to determine if there is room in the buffer for the EOD. BUFLM is never cleared and is set only at READCM or GETBUF.

For Teletype Terminals, TOUT contains the current pair of characters being output. A CM read is needed for TTY lines every two characters, and TOUT is set with the current pair at RD20. The first (or left) character is fetched from TOUT at STCH for output, while the second (or right) character is fetched at STC1. Since TOUT is never cleared after STC1 is executed, it will always contain the last pair of characters output. After the end of an output operation, TOUT would normally contain an EOB or EOD code. When TOUT is zero, a carriage return-line feed sequence should be in process, which was initiated by the output data.

IN:

The IN table contains the unprocessed input data from the multiplexor for each line. After the multiplexor is functioned for input, the input operations at F618 (for 6676s) and/or F118 (for 6671s) load the IN table. The contents of IN are not destroyed until the next input function. The high order 4 bits of IN are the port status bits, while the low 8 bits are the actual input data character if one is present (when the highest bit 4xxx is set).

The high order bit is checked for the presence of a valid input character at TLOOPA (for TTY lines) and at LOOP (for 200 USER lines). In SCAN at SC40 the "terminal ready" and "carrier on" status bits are checked prior to an output to determine that the data set is operational. Also, at STC4 the status is checked to detect if the data set on a TTY line is still operational. The 2^8 bit is the data rejected status bit, which is checked at TLOOPC (TTY) and at LOOP (200 USER). (See also the description of OTASK.)

OUT:

The OUT table contains the next data character with multiplexor control bits attached to be output to the terminal. After the multiplexor is functioned for output, the table OUT is output to the 6676 multiplexor at F610 and/or the 6671 multiplexor at F110. The next input is immediately read into the table IN. If the 2^8 bit in IN is clear, then the output word was accepted and another character must be prepared. Because the next TTY output character may not be available, OUT is cleared at TL00PC when the 2^8 bit is cleared to avoid having the last character repeat. If the next character for a 200 USER line cannot be prepared in time, then a SYNC word with output control bits $\{4026_8\}$ is placed in OUT.

OTASK:

The sequencing and preparation of all output to the terminals is performed by table driven tasks. The table OTASK contains the address of the next sequence of code {task} to be executed whenever the current output word {in OUT} is successfully written to the multiplexor and accepted {the 2^8 bit "data rejected" is clear in the corresponding entry in the IN table} by the multiplexor. Normally each task will store an output character with multiplexor control bits in OUT and reset OTASK to a task which would prepare the next output word.

For 200 USER lines, there are some tasks which do not generate output data which nevertheless use OTASK for sequencing {e.g., SCAN, SCAN1}. When CS is zero, the main processing loop for 6671 multiplexors at LOOP, will pass control directly to the OTASK sequence, without checking the port control bits in IN. During input, OTASK always contain REJECT, which is initially set at RESYNC. Therefore, during input operations the OTASK sequence actually provides the input timeout check. Refer to the output task flow diagram for the sequencing of each task.

For TTY lines, during an input operation, OTASK is always set to SCAN1, since no timeout is provided by the driver between characters. The subsystem can then terminate the input operation by clearing RD in the LRB.

ITASK:

The initial processing of all input from the terminals is performed by table driven tasks. The table ITASK contains the address of the next task to be executed whenever an input character is received. Normally each task will either test the input character or store it in to a PP buffer, and reset ITASK to process the next input character. The determination that a new input character has been received from the multiplexor is made at L00P {200 USER} and TL00PA {TTY}, by checking the high order {2^{11}} bit of IN. When this bit {data received} is set, the 8 low order bits of IN contain an input character.

For 200 USER terminals, during the output phase, ITASK contains NT {set at SC60}, so that if the ▽data received▽ status bit should be set, the output task specified by 0TASK would be executed. This is the normal procedure anyway, because if ▽data rejected▽ status is present, control goes to REJECT {from L00P} and the ▽data received▽ bit is not tested. Before the input task is entered, the 8 bits of data are saved in TEMP and a check is made for an ETX character. If no ETX, then WAIT is reset and the ITASK sequence is executed. If an ETX was received and detected at VAL, control passes to PETX which checks to see if the line is inputting. If not, control goes to NT and the output task. 0therwise RELBUF is called which checks the contents of ITASK for NEXT which indicates the ETX was expected. NEXT is placed in ITASK when

an ETX is the next character of interest and BUFIN is clear.

For TTY lines, ITASK always contains the address of an input processing sequence. During output, ITASK is set to TBRK to check for a break character. If a break is received, output is terminated and ITASK is set to RTCH. When the EOD is received and written to CM, ITASK is set to TLOOPE.

WAIT:

The WAIT table contains a clock for each line used for timeout purposes. For 200 USER terminals the data set provides the clocking for the line. However, the driver also counts the milliseconds between characters for each line to detect data set or multiplexor failures and lost or no messages returned from the terminal.

When each new output character is placed in OUT by an OTASK, control returns to NEXTL where WAIT is reset to the value CHWAIT. Once during each main cycle of the driver (i.e., at least every 3.3 milliseconds) at ENDMUX, the main 24 bit millisecond clock (CK) is updated and the number of whole milliseconds which have elapsed since the last calculation is stored in the D portion of the instruction at MSEC. If the "data rejected" status bit is set in the multiplexor input word (IN), then WAIT is reduced by MSEC. If, in addition, CHWAIT milliseconds have elapsed since OUT was last set, then a timeout error is assumed and control passes to TIMERR.

During the line turnaround phase of communications, WAIT is initiatized with MTAR, which is the installation parameter for the turnaround time (milleseconds) for 2-wire data set connections.

A similar timeout is performed during input, except that WAIT is reset at VAL when the new character is detected, just before control goes to the next input processing task. Note that each ITASK sequence returns to NEXT rather than NEXT1. Once each cycle, if no input character is ready, the control at LOOP goes to the OTASK sequence, which is always set to REJECT during input. Here WAIT is reduced by MSEC and the timeout test is made.

For TTY lines on a 6671 multiplexor, WAIT is used at INTX during initialization, to allow the multiplexor line disconnect code 6000_8 to complete. That is the only place where WAIT is used for timing purposes for TTY lines. Otherwise, WAIT is not used for TTY lines.

LSWA:

The LSWA table contains the relative CM address of the FWA of the Line Status Table for this line. Except for party lines, this address never changes from its initial value set during initialization. For the party line terminals connected to a given multiplexor port, LSWA is reset using the PLB from the current terminal, when the OTASK sequence SCAN is executed. Initialization sets the PLB for party line terminals such that driver control of them circles round robin fashion. Note that PLB is zero for non-party line terminals. LSWA is used by the routines RLSW, RLSWO, RLRW, WLSW, and WLRW to read and write the Line Status and Line Request words in the Line Status Table area in CM. It also is used in READCM and WRITCM to check that the message buffer is within the Line Status Table area. For RESPOND, the terminal number placed in the first byte of the CIRCSTACK word is computed from LSWA and LFWA in the routine GETCIR. Finally, in WRCM at WT37, LSWA is checked against CBB to insure that TTY data for the message buffer is stored within the Line Status Table. LSWA should never be zero.

12.2 Other Tables, Not One per Port

JLINE/JUNGW: {200 USER Terminals Only}

These two circular tables are both JUNCRT+1 words long. Together they form the queue of data to be processed by WRITCM. Entries are made to this queue only by QCMW. The queue has in and out indexes in the direct cells JNEXTW and JWRITE respectively. The range of values for JNEXTW and JWRITE are zero through JUNCRT. When the contents of JNEXTW and JWRITE are equal, the queue is empty and WRITCM will exit immediately when called.

JUNGW contains the LWA of the PP POOL buffer which is to be written to CM. JLINE contains the control information and line number associated with the buffer. The format of JLINE is as follows:

PB, NL, 0, LINE {9}

PB is the partial buffer flag. When PB=1, some data {up to 79 characters} is contained in the associated buffer, but no more data will follow. PB is set at RCLC39 when the E code is recognized. NL is the null buffer flag. When NL=1, no more data exists and JUNGW is not used. This request will not be queued {by incrementing JNEXTW} if the last buffer not written {data received} flag in WS is not set. The purpose of NL is to tell WRITCM to clear the last buffer not written flag only after all prior data has been processed to CM. When exactly 80 characters, or a multiple thereof, are input to CIRCSTACK, then the NULL buffer

will cause an EOD word to be written {for E code = E1 only}.

If both PB and NL are zero, then the buffer is full with 80 characters. All card data from the 224-1 card reader should be a multiple of 80 characters and the PB bit wouldn't be used. However, most messages from the display keyboard would not be exactly 80 characters and PB would be set.

When either PB or NL are set, WRITCM will clear BN in WS. Before WRITCM exits, JWRITE will be incremented.

If the driver is assembled with TTY not zero, then JUNCRT may be reduced below 2*JUN+1 words. The tables need to be long enough to queue all the POOL buffers available to CRT's and also one null buffer for each CRT line which has a buffer queued. As an example, if JUN=11, but TTY=5, then only 10 buffers could be queued and JUNCRT=21 would be sufficient. This table could not be set up as a line indexed table, since it may be necessary to queue more than one buffer per line {e.g., during a long storage move}.

JURD:

This circular table, TTY+1 words in length represents the queue of requests for TTY output data to be processed by RDCM. Entries are made to this queue only at QTYR. The queue has in and out pointers in the direct cells JUIN and JU0UT respectively. The range of values for JUIN and JU0UT are JURD through JURD+TTY. When the contents of JUIN and JU0UT are equal, the queue is empty and RDCM will exit immediately when called.

JURD contains the line number {from L} of the TTY terminal which needs more output data. The current pair of TTY output characters are stored in T0UT {see Table description} and when STC1 places the second {or right} character into 0UT, QTYR is called to increment JUIN and store the TTY line number in JURD.
RDCM then will detect that JUIN and JU0UT are not equal. JU0UT will be incremented and the line number will be fetched from JURD.

JUNGL:

This circular table, JUN+1 words in length, represents the queue of unused POOL buffers. These POOL buffers are 8 CM words (40 PP words) long. The queue has in and out pointers in the direct cells JEMPTY and JNEXT respectively. The range of values for JEMPTY and JNEXT are JUNGL through JUNGL+JUN. The direct cell JUNGLX contains the LWA+1 of the table (i.e., JUNGL+JUN+1).

JNEXT contains the location in JUNGL of the FWA of an available POOL buffer. If there are no POOL buffers available, then the entry in JUNGL at location JNEXT will be zero. When a buffer is taken from the POOL, JNEXT is incremented by one. The subroutine REQBUF is used to obtain POOL buffers. Only in REQBUF is JNEXT referenced. During initialization, JUNGL+1 is stored into JNEXT.

JEMPTY contains the location in JUNGL where a POOL buffer may be returned to the queue, after it is no longer needed. When a buffer is returned to the queue (POOL), JEMPTY is incremented by one and zeroes are stored at that location in JUNGL. The subroutine RETBUF is used to return POOL buffers. Only in RETBUF is JEMPTY referenced. During initialization, JUNGL is stored in JEMPTY and the first word of JUNGL is zero. The entries in JUNGL between JEMPTY and JNEXT are the FWA of POOL buffers currently in use.

For TTY lines, initialization obtains POOL buffers and cracks them into five 8 PP word buffers, one for each TTY line. Only enough buffers are taken for the actual number of TTY lines on the active

multiplexors. 1-5 TTY's would require one POOL buffer, 6-10 TTY's would require two POOL buffers, etc. Although these buffers are never returned, the scheme allows for the availability of the maximum space for buffers under varying multiplexor definitions.

For 200 USER lines, buffers are used only during actual transmission or while data is queued for WRITCM. During output READCM gets the buffer and fills it with data from CM. After the last character has been output, the buffer is returned at EJUNG. During input a buffer is requested at RCLB. When it is filled, it is queued for WRITCM. After the data has been written to CM the buffer is returned at WR35.

JNEXT should never equal JEMPTY, except when all the buffers are in use.

13.0 DIRECT CELL USAGE

The following is a description of the direct cell usage of the Driver:

Cells	Use
1-27	Scratch
30-74	Permanently assigned
75-77	PP register pointers

When the LSW and LRW are read from CM, they are always placed in cells 1-5 and 6-12, respectively. All of the permanently assigned cells above 27 are assigned tags with EQU's to D.XXX symbols, except for D.RA, D.FL, D.PPONE and D.CPAD, which are directly referenced.

The following cells are used for more than one purpose at different phases of the Driver's main cycle:

TO, FLAG, LAST, TEMP and FROM

All others are used only for one purpose.