6600

CHIPPEWA OPERATING SYSTEM

60124500	**Record of Revisions**
REVISION	NOTES
C	This printing obsoletes all previous editions.
(4-29-65)	

Pub. No. 60124500
April, 1965

Address comments concerning this manual to:

Control Data Corporation
Technical Publications Department
4201 North Lexington Avenue
St. Paul, Minnesota 55112

or use Comment Sheet located in the rear of this book.

TABLE OF CONTENTS

TABLE OF CONTENTS (Cont.)

APPENDIX A

OPERATING SYSTEM FLOW CHARTS

APPENDIX A

OPERATING SYSTEM FLOW CHARTS (Cont.)

APPENDIX A

OPERATING SYSTEM FLOW CHARTS (Cont.)

APPENDIX B

I/O SUBROUTINE FLOW CHARTS

APPENDIX C

SIM FLOW CHARTS

APPENDIX C

SIM FLOW CHARTS (Cont.)

APPENDIX C

SIM FLOW CHARTS (Cont.)

System Tape -

The system tape is loaded from the dead start panel via tape unit 50. This tape contains the operating system and the peripheral and central program libraries. Dead start panel settings are as follows.

```
01   1410
02   7305
03   0006
04   7505
05   7113
06   0000
07   7705
10   2000
11   7705
12   2020
13   7405
14   7105
```

The system tape is completely loaded at dead start time. All data is transferred to the channel 0 disk file. The tape is not referenced during system operation.

The system tape contains a single file of binary records in standard format. There is one file mark at the end of the data. Copies of the system tape may be made by COPYBF in the central library on either tape or cards. Data on the system tape is organized as follows. Each item is treated as one record.

```
Load package.
System display (DSD).
Monitor (MTR).
Central resident (CR).
Peripheral package 1LT.
Peripheral package 1TD.
Peripheral package 2PC.
Peripheral package DMP.
Peripheral package LBC.
Peripheral package LOC.
Peripheral package PBC.
Peripheral package DIS.
Peripheral package SIM.
(zero length record)
Central library (one record each package)
```

Dead Start Load -

The dead start load of the system tape begins with the dead start panel settings on peripheral processor 0. Processor 0 transfers a short program to processor 5 which reads the first record from tape 50 on channel 5. This first record contains the peripheral resident program which is common to all processors and a transient package which loads the remainder of the system tape.

Processor 5 transfers the peripheral resident program to the other nine processors over their respective channels. Processor 5 then reads the next record from the system tape (DSD) and transfers it to processor 9. Processor 5 then reads the third record from the system tape (MTR) and transfers it to processor 0. Processor 5 then reads the next record from the system tape and transfers it to central storage. Processor 5 then disconnects all channels other than 5. This action starts execution in all ten processors. Processor 0 is permanently assigned the job of system monitor. Processor 9 is permanently assigned the job of system display. The other eight processors form a pool for temporary assignment of I/O packages.

Processor 5 reads the system tape to a zero length record. Each record prior to the zero length record is transferred to the disk file on channel 0 as a peripheral package. An entry is made in the Peripheral Library Directory (PLD) as each package is transferred.

Processor 5 then reads the remainder of the system tape and transfers each record to the channel zero disk file as a central processor package. An entry is made in the Central Library Directory (CLD) for each package transferred. Processor 5 then idles in the resident peripheral program until action is requested as a pool PP.

Equipment Numbers -

Each peripheral equipment is assigned a two digit octal number to uniquely identify the equipment for job assignments. This equipment number is the same as the relative location of the equipment data in the equipment status table. Numbers are assigned as follows.

00 - Channel 0 disk file.
01 - Channel 1 disk file.
02 - Channel 2 disk file.
04 - Channel 4 card reader.
05 - Channel 12 card reader.
06 - Channel 13 card punch.
07 - Channel 13 line printer.
10 - Channel 10 display.
11 - Channel 11 display.
30 - Channel 3 one inch tape.
31 - Channel 3 one inch tape.
32 - Channel 3 one inch tape.
33 - Channel 3 one inch tape.
40 - Channel 4 half inch tape.
41 - Channel 4 half inch tape.
42 - Channel 4 half inch tape.
43 - Channel 4 half inch tape.
50 - Channel 5 half inch tape.
51 - Channel 5 half inch tape.
52 - Channel 5 half inch tape.
53 - Channel 5 half inch tape.
60 - Channel 6 one inch tape.
61 - Channel 6 one inch tape.
62 - Channel 6 one inch tape.
63 - Channel 6 one inch tape.
70 - Channel 7 one inch tape.
71 - Channel 7 one inch tape.
72 - Channel 7 one inch tape.
73 - Channel 7 one inch tape.

Central Storage Special Addresses -

Central storage addresses 0000 thru 7777 are reserved for tables and communication areas for the operating system. Special fixed storage locations are as follows.

0000 Zero
0001 RPL, , , , .
0002 PLD, limit, , , , .
0003 DFB, input, output, limit, .
0004 FNT, limit, , , .
0005 EST, limit, , , .
0006 RSL, limit, , , .
0007 CLD, limit, , , .
0010 TRT0, last track, , G0 to G3, G4 to G7.
0011 TRT1, last track, , G0 to G3, G4 to G7.
0012 TRT2, last track, , G0 to G3, G4 to G7.
0014 Monitor step control.
0015/0017 Channel status table.
0020 CP zero status.
0023 idle time for central processor.
0024 idle time for peripheral processors.
0026 XJ address for simulator.
0027 P address for simulator.
0030/0037 time and date line.
0040/0052 starting time counts.
0055 temporary storage for monitor.
0056/0057 CP stack indicators.
0060 PP1 communication area.
0070 PP2 communication area.
0100 PP3 communication area.
0110 PP4 communication area.
0120 PP5 communication area.
0130 PP6 communication area.
0140 PP7 communication area.
0150 PP8 communication area.
0160 PP9 communication area.
0170 PP0 communication area.
0200 Control point one area.
0400 Control point two area.
0600 Control point three area.
1000 Control point four area.
1200 Control point five area.
1400 Control point six area.
1600 Control point seven area.

Control Point Area Information -

There are seven control points. Each occupies 200 octal locations and includes the following information.

000/017 exchange package
020/022 status information
023 central processor running time.
024 peripheral processor running time
025 PP recall input register
026 sense lights and switches
027 equipment assignments
030/037 last dayfile message.
040/177 control statement buffer.

Word 020 -

(12) status byte (W, X, 1, 2, 3, 4, 5, 6, 7, 8, 9, 0)
(12) error flag.
(12) storage move flag.
(12) reference address (hundreds).
(12) field length (hundreds).

Word 021 -

(48) job name in display code.
(12) next statement address.

Word 022 -

(12) priority.
(12) message count.
(12) tract count.
(12) time limit (8 sec increments).
(12) operator assigned equipment.

Word 023/024 -

(24) not used.
(24) second count.
(12) millisecond count.

23 - cp time
24 - pp time

Equipment Status Table -

The equipment status table contains a one word entry for each physical equipment in the system. Format is as follows.

(12) Control point address.
(12) Channel number.
(12) Synchronizer and unit number.
(12) Equipment type.
(12) Not used.

Equipment type designations are as follows. (display code)

(DA) Channel zero disk cabinet.
(DB) Channel one disk cabinet.
(DC) Channel two disk cabinet.
(DS) Display console.
(CP) Card punch.
(CR) Card reader.
(LP) Line printer.
(MT) 607 magnetic tape.
(WT) 626 magnetic tape.

The upper bit of the equipment type designator is used as an interlock for equipment which is turned off. This bit is set when the equipment is not available and cleared when the equipment is available.

Channel Status Table -

The channel status table occupies 3 words (0015/0017) in resident central storage. These three words are read into monitor peripheral storage for updating and the first 12 bytes in these three words are then associated with the 12 I/O channels. Two psuedo-channels (octal 14 and 15) are used for access to the FNT/FST entries. A cleared byte indicates the channel is not assigned. A processor number indicates the channel assignment.

File Name Table / File Status Table -

The name and status of all system files are kept in common table structure. This table contains two words per entry. The first word identifies the file and the second word records the status of the file. Formats for each two word entry is as follows.

First Word.
(42) file name in display code.
(6) priority number.
(6) not used.
(3) file type (0 = input, 1 = output, 2 = common, 3 = local).
(3) control point assignment.

Second Word. (disk file)
(12) equipment number.
(12) beginning track.
(12) current track.
(12) current sector.
(12) last buffer status.

Second Word. (tape file)
(12) equipment number.
(24) last block number.
(12) not used.
(12) last buffer status.

Second Word. (card file)
(12) equipment number.
(24) card count this record.
(12) end of job flag.
(12) last buffer status.

The buffer status byte in the file status word serves as an interlock for peripheral processor access to the corresponding file data. The last buffer status has an even numerical value while a peripheral package is altering the file data. The last buffer status has an odd value when no peripheral action is involved with the file. Pseudo-channel 14 is requested for access to an existing file status. Pseudo-channel 15 is requested for access to enter a new file name in the table.

Resident Peripheral Library -

Central storage addresses 10000 thru 13777 are reserved for resident peripheral packages. Each package begins with a control word in the following format.

(42) package name in display code.
(18) package length.

The last package is followed by a blank word.

Peripheral Library Directory -

A directory of peripheral packages located in the disk file library is kept in resident central storage. Each entry in the directory requires one central word. Format of this word is as follows:

(42) package name in display code.
(6) sector number.
(12) track number.

Resident Subroutine Library -

Central program subroutines which are most frequently used are kept in resident central storage. A program calling for these subroutines may read them directly from this RSL (Resident Subroutine Library) area rather than from the disk file library. Each subroutine in the RSL is preceded by a one word tag in the following format.

(42) package name in display code.
(18) package length.

The last package is followed by a blank word.

Central Library Directory -

Central programs and subroutines which are not stored in the RSL are stored in a disk file library. A directory of these programs and subroutines is kept in central storage (CLD). Each entry in this directory consists of one central word in the following format.

(42) package name in display code.
(6) sector number.
(12) track number.

Disk File Organization -

Programs, alpha data files, and binary data files are stored in the disk file system in a common structure. The addressing mechanism and file marking system are the same for all types of files.

The physical disk structure of cabinets, head groups, tracks, and sectors, is modified slightly in system addressing. The basic unit for purposes of reserving storage is a half track. This half track unit consists of either the odd or the even numbered sectors in a physical track within a head group. This modification is intended to provide minimum access for a single processor streaming data from a disk record.

A disk file in the logical sense is a named half track (with continuation half tracks) in a single disk cabinet. A disk file may be of any length within the limits of the storage capacity of the disk cabinet. There are 2048 half tracks in a disk cabinet which may be grouped into named files of any length. Each named file must begin at the first sector of a half track.

Each sector in a file contains two control bytes in addition to the 320 data bytes in a sector. The first of these control bytes designates the location of the next sector in the named file. The second control byte designates the number of central words in the current sector. These two control bytes are the first two bytes in a sector.

The first control byte in a disk file sector is coded in one of two formats. If the next sector is in the same half track, the control byte is coded with the next sector number (0000 thru 0077). If the next sector in the named track is the beginning of a new half track, the control byte is coded for the next half track number. (4000 thru 7777). The format for this control byte is as follows.

(1 bit) always set.
(7 bits) track number.
(1 bit) odd/even track.
(3 bits) head group number.

The control bytes for the last sector in a named file are both zero. This sector is read as a file mark. No data can appear in a disk file after the first file mark.

Track Reservation Tables -

A 2048 bit reservation table is maintained in resident central storage for each disk file cabinet. One bit in the table corresponds to each half track in the disk cabinet. Each table occupies 64 central words and uses the lowest order 32 bits of each word. The highest order 7 bits of the track number specify table address and the lower order 5 bits of the track number specify bit position in the word.

Alpha Files (Tape, Disk) -

Alpha files are stored in packed display code for disk storage and magnetic tape storage. Each line of alphanumeric characters begins at the first byte of a central word and continues two characters per byte to the end of a line of code. The line of coding may be of any length. The last central word used in a line of coding is completed with cleared bytes. A cleared byte is the end of line designation.

Punched Card Formats -

Alpha files - Hollerith code - 80 columns.

Binary files - 15 words per card.
- 7, 9 punch column one.
- word count in column one row 0,1,2,3.
- correction punch in column one row 4.
- check sum modulo 4095 in column two.

Record separator - 7,8,9 punch column one.

File separator - 6,7,8,9 punch column one.

Resident Peripheral Program -

This program is loaded in each peripheral processor on dead start and is not altered throughout execution of the system. The resident program occupies peripheral addresses 0100 thru 0771. Temporary storage addresses 0000 thru 0017 are used by the resident program. These locations may also be used by a called peripheral package as no continuity is required between executions of the resident program.

Peripheral addresses 0075 thru 0077 are constants for a processor and should not be altered by a peripheral package. These addresses contain the following information.

0075 processor input register address
0076 processor output register address
0077 processor message buffer address

Peripheral Communication Areas -

Central storage addresses 0060 thru 0177 are used for peripheral processor communication. Each processor is assigned a block of eight words. Format within a processor area is as follows.

(one word) processor input register
(one word) processor output register
(six words) message buffer

The processor input register is cleared when the processor is idling. The monitor processor enters a control word in the input register to call a transient peripheral package to the processor. The resident peripheral program senses the input register entry and searches the RPL (Resident Peripheral Library) for the peripheral package named in the input register. If the named package is not in the RPL the resident program then searches the PLD (Peripheral Library Directory) for the named package. When the package has been located the resident program reads the package into peripheral storage beginning at peripheral address 1000. The resident program then jumps to peripheral address 1000 to begin execution of the transient package.

Format for the input register control word is as follows.

(18 bits) package name in display code
(6 bits) control point number
(36 bits) arguments

The processor output register is entered by a transient peripheral package to communicate with the peripheral monitor. The monitor continually scans the processor output registers and interprets the requests from the transient peripheral packages. The message buffer is used for those monitor requests in which the output register is not sufficiently large. The monitor program clears a processor output register when the requested function has been performed.

The processor input register is not altered during the execution of a transient peripheral package. Upon completion of the transient package a request to the monitor clears the input register and the transient package exits to the resident peripheral program (address 100).

Monitor Calls From Peripheral Processors -

The following control words are interpreted by the monitor processor when entered in a peripheral output register.

0001, 0000,0000,0000,0000 Dayfile message.
The monitor enters a message in the dayfile consisting of a time tag, a job tag, and the contents of the peripheral processor message buffer. The message buffer should contain one line in packed display code.

0002,00NN,0000,0000,0000 Request channel NN.
Assign channel NN to the requesting processor as soon as the channel is available.

0003,00NN,0000,0000,0000 Drop channel NN.
Release the assignment of channel NN from the requesting processor.

0004,0000,0000,0000,0000 Assign PP time.
Update the pp running time at the control point for the requesting processor.

0005,0000,0000,0000,0000 Monitor step control
Shift monitor control to step mode using central address 0014 for control.

0006,00NN,0000,0000,0000 Request track.
Request any unassigned track from track reservation table NN (10,11,12). If a track is available the track number is entered in the first byte of the process message buffer. If a track is not available a cleared byte is returned.

0007,00NN,TTTT,0000,0000 Drop track.
Drop track TTTT from track reservation table NN (10,11,12)

0010,NNNN,0000,0000,0000 Request storage.
Assign NNNN hundred octal words of central storage to the control point of the requesting processor. If a conflict exists the monitor may not assign the requested storage. The requesting processor must sense the control point status to verify the actual assignment. A pause function (17) should preceed further requests for storage.

0011,0000,0000,0000,0000 Release PP.
Release the peripheral processor from the control point. The monitor clears the processor input register before resuming this function.

0013,0000,0000,0000 Abort control point.

Abort the job associated with the requesting processor. The requesting processor is responsible for an explanatory message in the dayfile. The monitor clears the PP input register before resuming this function.

0014,TTTT,0000,0000,0000 Time limit.

Enter a job time limit at the associated control point TTTT increments of time. Each time increment is 8 seconds.

0015,0000,0000,0000,0000 Request central processor.

Set the central waiting flag (W) at the associated control point. Search for job priorities to initiate central processor action.

0016,0000,0000,0000,0000 Drop central processor.

Drop central processor execution for the job at the associated control point.

0017,0000,0000,0000,0000 Pause for relocation.

This function allows the monitor to move central storage data for the associated job. The requesting processor should check the reference address for the control point after resume from this function.

0020,0000,0000,0000,0000 Request PP.

This function requests the initiation of another peripheral processor. The first word of the message buffer associated with the requesting processor contains the input register data for the new PP. This function is resumed with the PP input register address in the first byte of the message buffer. If no PP is available a blank byte is returned.

0021,0000,0000,0000,0000 Recall central processor.

Restart the central program associated with the requesting PP only if the central recall flag (X) is set.

0022,NNNN,0000,0000,0000 Request equipment

Search the equipment status table for an equipment of type NNNN and assign this equipment to the control point. The monitor responds with the equipment number in the first byte of the PP message buffer if an equipment of the proper type is available. The monitor responds with a blank byte if the equipment is not available.

0023,00NN,0000,0000,0000 Drop equipment.

Drop equipment number NN from the associated control point.

0024,00NN,0000,0000,0000 Request priority.

Assign priority number NN to the control point.

0025,000N,0000,0000,0000 Error exit mode.
Assign exit mode N to the central program at the associated control point.

0027,0000,0000,0000,0000 Toggle simulator.
Toggle the status of the central processor and simulator. The simulator is called from the peripheral library or dropped as required.

0030,000N,0000,0000,0000 Operator drop.
Drop the job at control point N by setting the operator drop error flag.

0031,00NN,0000,0000,0000 Equipment on.
Clear the lockout bit for equipment NN in the equipment status table.

0032,00NN,0000,0000,0000 Equipment off.
Set the lockout bit for equipment NN in the equipment status table.

0033,00NN,000P,0000,0000 Assign equipment.
Assign equipment number NN to control point P and enter the number NN in the control point area for operator assignment.

Peripheral Program Formats -

Binary versions of peripheral programs contain data in the first word of the program code which controls loading of the package. All peripheral packages in the RPL (Resident Peripheral Library) and PLD (Peripheral Library Directory) are overlays which are called by the resident peripheral programs. These overlays are called once and executed and then discarded when the next overlay is called. All overlays are coded for a particular beginning address in peripheral storage. Overlays which begin at address 1000 in peripheral storage are the basic transient peripheral packages. These transient packages may call equipment driver overlays which begin at peripheral address 2000. The various types of peripheral overlays are identified by name only in the calling formats. Those basic transient packages which may be called by a central program are assigned names beginning with a letter. Those packages which are equipment driver overlays or are special operating system packages are assigned names beginning with a numeral.

The first word of a peripheral package contains the following information.

(18 bits) Package name in display code.
(30 bits) Not used.
(12 bits) Package length in central words.

Resident Peripheral Library -

The following peripheral overlays are stored in the resident peripheral library.

1AJ - Advance Job.

1BJ - Begin Job.

1DJ - Phase three print.

1LJ - Phase one card load.

2BD - Backspace disk.

2BP - Buffer parameters.

2BT - Backspace tape.

2DF - Drop file.

2DT - Drop tracks.

2EF - Error flag.

2LP - Line printer

2RC - Read cards.

2RD - Read disk.

2RT - Read tape.

2SD - Search dayfile.

2TJ - Translate job card.

2TS - Translate statement.

2WD - Write disk.

2WT - Write tape.

CIO - Circular I/O.

CLL - Call library.

EXU - Execute.

MSG - Message for dayfile.

Peripheral Library Directory -

The following peripheral overlays are listed in the peripheral library directory.

1LT - Phase one tape load.

1TD - Phase three tape dump.

2PC - Punch cards.

DMP - Dump storage.

LBC - Load binary cards.

LOC - Load octal cards.

PBC - Punch binary cards.

DIS - Display job.

SIM - CPU simulator

DSD - System Display - Channel 10

This package is loaded in processor 9 on dead start and remains there unaltered throughout execution of the system. This package provides an overall status display for all currently running jobs. The keyboard is used to initiate and control equipment assignment and job progress.

The two console screens may be assigned any combination of two of the following displays.

A - dayfile
B - job status
C - data storage (5 groups of 4 octal digits)
D - data storage (5 groups of 4 octal digits)
E - data storage (5 groups of 4 octal digits)
F - program storage (4 groups of 5 octal digits)
G - program storage (4 groups of 5 octal digits)
H - job backlog

As an example, the following keyboard entry would select H display on the left screen and B display on the right screen.

HB.

The storage displays C through G consist of 4 fields of 8 words each. The fields are designated 0,1,2,3 and may be individually changed to any 8 word group of central storage. The following keyboard entry would change the D display, field 2, to display central storage addresses 001020 through 001027.

D2, 1020.

If D4,1020 were used in the above example the D display would be changed to 4 consecutive fields beginning with address 001020.

Any word in central storage may be modified from this console. An entry in central storage is made in the following format.

1022, 2100 1000 0000 0111 2347.

Spacing is not important. Leading zeros may be dropped in both address and data word. All entries are octal and the digits are right adjusted.

AUTO.

This keyboard entry may be used after a dead start from the system tape to select a mode for automatic job processing. This mode assumes job inputs from punched cards and printer output.

STEP.

This keyboard entry selects a step mode for the operating system monitor. Each processor request is stepped by depressing the keyboard space bar. High speed operation may be resumed by entering a period on the keyboard.

SIM.

This keyboard entry calls a central processor simulator package from the disk file library to a peripheral processor. Subsequent system operation is with a simulated central processor. The simulator may be dropped by repeating the keyboard entry SIM. Repeated entries will toggle the system between simulated and real central processor.

ON42.

This keyboard entry is an example of format for designating to the operating system that a particular equipment is working and ready for automatic monitor assignment to any job requesting this type of equipment. The equipment is designated by a two digit octal number corresponding to the position of the equipment in the equipment status table.

OFF42.

This keyboard entry is an example of format for removing a particular equipment from automatic monitor assignment. The equipment is designated by a two octal digit number corresponding to the position of the equipment in the equipment status table.

TIME. 12.10.03, March 12, 1965.

This keyboard entry permits the operator to enter the correct time and date.

1. READ.

This keyboard entry is an example of format for selecting a punched card job input package. The leading digit designates which control point is to be used for the input package. Jobs are read from punched cards as long as cards are entered in the card reader. All job data is stored in disk files, and the jobs are executed according to the priorities on the job cards.

2. PRINT.

This keyboard entry is an example of format for selecting a printer output package. The leading digit designates which control point is to be used for the print package. Job output files are printed in the order of job priorities.

3. NEXT.

This keyboard entry is an example of format for selecting a control point, as designated by the leading digit, to process job executions. Jobs are called one at a time in order of priority.

4. GO.

This keyboard entry is an example of format for restarting a job which has come to a pause statement. The leading digit must correspond to the control point with the pause statement displayed.

4. DIS.

This keyboard entry is an example of format for calling a job display package (DIS) to a control point as designated by the leading octal digit.

4. DROP.

This format drops the job at the designated control point.

4. ASSIGN 53.

This format assigns a particular equipment (53) to a particular control point (4). This method of equipment assignment should be used only when the job is waiting for the equipment assignment as indicated by a REQUEST statement display.

7. LOAD.

This keyboard entry selects a magnetic tape job loader package at the designated control point. The package will request operator assignment of a specific tape unit. The tape read on this unit must have input jobs stacked to a double file mark. Such a tape may be prepared by copying card to tape with the library program COPY.

5. DUMP.

This keyboard entry selects a magnetic tape dump package at the designated control point. The package will request operator assignment of a specific tape unit. The current backlog of completed output files will be dumped on this tape in the order of job priority. A file mark will be recorded and the tape backspaced to allow repeated dumps on the same tape. The tape may then be listed off line with a tape to printer program.

4. ONSW2.

This keyboard format allows setting of the six sense switches for Fortran programs during execution.

4. OFFSW4.

This keyboard format allows clearing of the six sense switches for Fortran programs during execution.

5. DCN11.

This format allows the operator to manually disconnect any I/O channel. The channel number is octal.

5. FCN10.

This format allows the operator to manually enter a zero function on any I/O channel. The channel number is octal.

DIS - Job Display

This package may be called at any time during the execution of a job. This package, when called, stops further automatic advance of the job control cards. The display provided in this package covers only data pertaining to the particular job. The keyboard is used to advance the job control cards and provide any combination of two of the following displays.

A - dayfile
B - job status
C - data storage (5 groups of 4 octal digits)
D - data storage (5 groups of 4 octal digits)
E - data storage (5 groups of 4 octal digits)
F - program storage (4 groups of 5 octal digits)
G - program storage (4 groups of 5 octal digits)

As an example, the following keyboard entry would select A display on the left screen and B display on the right screen.

AB.

The storage displays C through G consist of 4 fields of 8 words each. The fields are designated 0,1,2,3 and may be individually changed to any 8 word group of central storage. The following keyboard entry would change the D display field 2 to display central storage addresses 001020 through 001027.

D2, 1020.

If D4, 1020 were used in the above example the D display would be changed to 4 consecutive fields beginning with address 001020.

Any word in central storage which lies within the field length of this job may be modified from this console. All addresses are relative to the job reference address. An entry in central storage is made in the following format.

1022, 2300 7775 1143 0000 0022.

Spacing is not important. Leading zeros may be dropped in both address and data word. All entries are octal and the digits are right adjusted.

The following keyboard entry formats may be used to alter the contents of the exchange package for the job displayed. If the program is running the operation will stop before the entry is made. The numerical values in the examples below are for illustrating format only.

ENP, 12345.

Enter the value 12345 in the exchange package for the next program address.

ENA3, 665000.

Enter the value 665000 in the exchange package for the A3 register.

ENB2, 44.

Enter the value 44 in the exchange package for the B2 register.

ENX5, 2223 4000 0000 0000 0200.

Enter the indicated value in the exchange package for the X5 register.

ENEM, 7.

Enter the value 7 in the exchange package for the exit mode.

ENFL, 10000.

Enter a field length of 10000 for the exchange package.

ONSW3.

This keyboard format allows the operator to set a Fortran sense switch during execution.

OFFSW5.

This keyboard format allows the operator to clear a Fortran sense switch during execution.

The following keyboard entry formats may be used to change the status of a job. The numberical values are for illustration of format only.

ENTL, 200.

Enter a time limit of 200 octal seconds.

ENPR, 5.

Enter a job priority of 5.

DCP.

Drop the central processor and display the exchange area.

RCP.

Request the central processor and begin execution at the next program address.

BKP, 44300.

Breakpoint to address 44300 in the program. This entry begins central processor execution of the program at the next program address and stops execution when the program register reaches address 44300. The contents of address 44300 is cleared while the program is running and is restored when the break point address is reached.

RNS.

Read the next control statement and begin execution.

RSS.

Read the next control statement and stop before execution.

DROP.

Drop the display package and continue execution of the remaining control statements.

HOLD.

Drop the display for later recall by the system display console. This entry holds the job at its present status.

GO.

This keyboard entry restarts a program that has stopped at a pause statement.

ENS. XXXXXXXXXXXXXX.

This keyboard format allows the entry of any control statement XXXXXXXXXXXXXX as if it had been entered on a control card.

DMP(200,330)

This format dumps storage addresses 200 to 330 in the output file.

DMP(400)

This format dumps storage from the reference address to address 400.

DMP.

This format dumps the exchange package in the output file.

CIO Circular Buffer I/O Package

Interface with central program -

This peripheral package is called by entering CIO in display code left adjusted in RA + 1. The lowest order 18 bits of RA + 1 contain the relative storage address (BA) of the circular buffer parameters. These parameters occupy five central words beginning at BA.

BA.	(42 bits)	File name in display code left adjusted.
	(12 bits)	not used.
	(6 bits)	buffer status.
BA + 1.	(42 bits)	not used.
	(18 bits)	FIRST. First address of circular buffer.
BA + 2.	(42 bits)	not used.
	(18 bits)	IN. Next buffer input address.
BA + 3.	(42 bits)	not used.
	(18 bits)	OUT. Next buffer output address.
BA + 4.	(42 bits)	not used.
	(18 bits)	LIMIT. Last buffer address + 1.

If IN = OUT the buffer is empty. Maximum buffer capacity is LIMIT - FIRST - 1. IN and OUT may equal FIRST but may not equal LIMIT.

The six bit buffer status indicates the mode of the buffer and provides an interlock for peripheral package activity. The buffer status has an even numerical value when CIO is called. The buffer status is set to an odd value when the peripheral package has completed the I/O function. Specific status indicators may be obtained by combining the octal digits in the table below.

0X	not used	X0	request coded read
1X	buffer I/O	X1	complete coded read
2X	end record	X2	request binary read
3X	file mark	X3	complete binary read
4X	backspace	X4	request coded write
5X	rewind	X5	complete coded write
6X	rewind unload	X6	request binary write
7X	not used	X7	complete binary write

CIO treatment of status codes

(1) Input (001,0X0)
File is read into circular buffer until buffer is filled (001,0X1) or until end record (010,0X1) or file mark (011,0X1). The mode bit in this function request is ignored and mode is determined by the input medium. A file mark response should never occur with data.

(2) Output (001,1X0)
Data in circular buffer is recorded on file for as many complete physical records as available data. No partial end record is made. (001,1X1).

(3) End Record (010,1X0)
Data in circular buffer is recorded on file including a short physical record to mark end of logical record. The last physical record may be of zero length. IN = OUT = FIRST. (010,1X1)

(4) File Mark (011,1X0)
(a) Data in circular buffer; or
(b) Last buffer status (001,1XX)
Data in circular buffer is recorded on file including a short physical record to mark end of logical record. Then a file mark is recorded. IN = OUT = FIRST. (011,1X1)
(c) All others
A file mark is recorded. IN = OUT = FIRST. (011,1X1)

(5) Backspace Binary (100,X10)
Back file to end of last record. A file mark is considered as a record in this case. IN = OUT = FIRST. (100,011)

(6) Backspace Coded (100,X00)
Back file one coded line. A file mark is considered as a coded line in this case. The last physical record will be left in the buffer beginning at FIRST. IN and OUT will be adjusted for a one line backspace. (100,001)

(7) Rewind (101,XX0)
Rewind the file. IN = OUT = FIRST. (101,0X1)

(8) Rewind Unload (110,XX0)
Rewind and unload the file. IN = OUT = FIRST. (110,0X1)

CIO special properties.

(1) The central program must complete writing a record before requesting a backspace, rewind, or mode change.

(2) The central program need not complete writing a record before requesting a file mark.

(3) The central program must complete reading a record before beginning output data to the buffer.

(4) Binary tapes and coded one inch tapes record 1000 octal word physical records in odd parity. No special control words are added. A zero length physical record is generated by recording a partial word (4 bytes). Coded one inch tapes use packed display code with short word separators.

(5) Coded half inch tapes record 120 character BCD code in even parity.

(6) A one inch tape with mixed binary and coded records presents problems if a backspace is used to cross a mode boundary.

(7) A problem exists in mode change from coded input to output on one inch tape if the file is positioned between two lines of code. No problem exists if the file is positioned before or after a file mark. No problem exists in mode change on binary files.

(8) A disk file connot be substituted for a tape file if the file has multiple file marks or has data recorded after a file mark.

DMP - Dump Storage.

This peripheral package may be called from a control card or from a DIS console. An octal dump is entered in the OUTPUT file with the central storage address and one data word per line. There are several format variations for calling this package.

DMP.

This format dumps the exchange area into the OUTPUT file.

DMP, 3400.

This format dumps from the reference address to the argument address.

DMP (4000,6000)

This format dumps from the first argument address to the second argument address.

LBC - Load Binary Corrections.

This peripheral package may be called from a control card or from a DIS console. Binary corrections are read from the INPUT file and are entered in central storage. If an argument is used in the package call, the binary cards are loaded beginning with that address. If no argument is used, the binary cards are loaded beginning at the reference address. Only one record is read from the INPUT file. If more than one block of data is to be entered more calls must be made.

The two types of format for this package call are as follows.

LBC.

LBC, 2300.

LOC - Load Octal Corrections.

This peripheral package may be called from a control card or from a DIS console. Octal corrections are read from the INPUT file and are entered in central storage. The octal cards which are used for these corrections must be in the following format.

```
23001 45020 04000 00042 00044
```

The address must begin in column one. Leading zeros may be dropped in the address. The data word must not begin before column 7. Spacing in the data word is not important but the word must contain 20 digits.

Several formats are allowed in making this package call. The simplest is the following.

```
LOC.
```

This call reads all of the correction cards in the next INPUT file record and modifes central storage accordingly.

```
LOC, 1000.
```

This call clears central storage from the reference address to the argument address. The correction cards are then read from the INPUT file.

```
LOC(2022, 3465)
```

This call clears central storage from the first argument address to the second argument address. The correction cards are then read from the INPUT file.

This package may be called to clear storage only by providing an empty record in the INPUT file.

PBC - Punch Binary Cards.

This peripheral package may be called from a control card or from a DIS console. This package punches a deck of binary cards directly from central storage. Storage is not modified by this operation.

Several formats may be used in calling this peripheral package.

PBC, 2000.

This format causes a binary deck to be punched from the reference address to the argument address.

PBC(2000,3000)

This format punches a binary deck from the first argument address to the second argument address.

PBC.

This format punches a binary deck using the first word in central storage as a control word for deck length. The deck always begins at the reference address and terminates one address less than that indicated in the lower 18 bits of the first word. This format may be used for punching any central or peripheral program in standard format.

CLL - Call Central Overlay

This peripheral package calls one or more central overlays to addresses supplied by the calling program. The lowest order 18 bits of the call word in RA+1 contain the storage address (BA) for arguments. The first two words in the argument region contain beginning and limit addresses for the overlays as a group. A list of the overlay names begins in BA+2 and continues to a cleared word.

BA.	(42 bits)	not used.
	(18 bits)	beginning address for first overlay.
BA+1.	(42 bits)	not used.
	(18 bits)	limit address for group of overlays.
BA+2.	(42 bits)	name of first overlay in display code.
	(18 bits)	beginning address of overlay (inserted by CLL).
BA+3.	(42 bits)	name of second overlay in display code.
	(18 bits)	beginning address of overlay (inserted by CLL).

etc.

CLL searches for the overlays in the order named first in the Resident Subroutine Library (RSL), then in the Central Library Directory (CLD), and then in the assigned job files. As a named overlay is located it is copied into central storage at the next available address. This address is then entered into the argument region with the name of the overlay. If an overlay cannot be located the address is left blank in the argument region. If an overlay exceeds the limit address a 777777 is entered as the argument address. The last overlay is followed by a cleared word. CLL then clears (BA) to indicate completion of the call.

EXU - Call and Execute.

This peripheral package is called by entering EXU in display code left adjusted in RA + 1. The lowest order 18 bits of the call word indicates the central storage address for the argument. The argument consists of the name of a central program to be called and executed. This call destroys the calling program and completely replaces it with the called program.

MSG - Dayfile Message.

This peripheral package is called by entering MSG in display code left adjusted in RA + 1. The lowest order 18 bits of the call word indicates the central storage address for the argument. The argument consists of a sequence of words in display code which are to be entered in the dayfile. The time and job name are automatically inserted before the argument data. The data is terminated by a cleared byte.

Job Control Cards -

A job consists of one or more central programs which are executed with common data files. Job control cards are used to identify the programs and data files and sequence the program executions.

Each job must begin with a job card. This card names the job and provides priority, time limit, and field length information for the job. An example of a job card is as follows.

JOB237, 6, 400, 27000.

In this example the job name is JOB237. Priority is 6. Time limit is 400 octal seconds. Field length is 27000 octal words. Spaces are not important in the job card format except that the job name must begin in column one. The job name must begin with a letter and may have up to 7 characters. The priority is a single decimal digit. A higher priority is represented by a larger number. The time limit may consist of up to 5 octal digits. The octal value in hundreds is approximately the time in minutes. The field length must be in hundreds and cannot exceed 360,000.

The rest of the job control cards must follow the job card in the order of execution. A record separator follows the last job control card. Program and data cards then follow in the order of input requests.

File Names -

All input and output actions of a central program must involve a named file. This file may be a disk file, a magnetic tape file, a punched card file, or a printer output file. The physical unit associated with a file name is controlled by the job control cards and is not a function of the central program coding directly. The operating system provides a common interface between the central program and the peripheral programs which drive the peripheral equipments.

File names must begin with a letter and may have up to 7 characters. There are two special names which are implied with each job. These names are INPUT and OUTPUT. The name INPUT refers to the file from which the job cards were read. This file is the result of loading the card deck for the job. The name OUTPUT refers to the file which ends in printed copy at the end of the job. These two names must be avoided in assigning names to temporary files and other I/O files.

Program Calls -

A program is called from a file or the program library by a control card in the following format.

SORT(T1, T2, XLF)

In this example the name of the program is SORT. This program may have been loaded from a file specifically associated with this job, or it may have been compiled from a fortran deck in the input file, or it may be in the general program library. All files associated with the job are searched for this particular name. If no job file has this name the program library is searched for a program of this name. When the program has been located it is read into central core storage. The arguments within the parenthesis are then entered in the program beginning at address 000002. These arguments are names for other files associated with the job and are entered in the forepart of the program in display code left adjusted in each word. The number of arguments must agree with number provided in the program. The execution of the program is initiated at a program address specified in the forepart of the program. (see program format)

If a file name appears for the first time as the argument in a program call it is assumed to be a disk file and is assigned to disk storage at that time.

Special Control Card Formats -

The following control card formats are recognized as special cases and must not be used for the names of programs.

ASSIGN MT, AFILE.

This format assigns any available peripheral unit of type MT to a file name AFILE. This must be the first appearance of AFILE.

COMMON BFILE.

This format causes a search for the name BFILE in the list of common files. If a common file is located with this name, and if the file is not being used by another job, it is assigned to this job until dropped. If the file is being used by another job or does not yet appear in the list of common files, this job must wait until the file is available.

If BFILE already appears as a local file name for the job, this format transfers that file to common status and is available to other jobs when dropped by this job.

RELEASE BFILE.

This format causes a common file named BFILE currently assigned to this job to be dropped from common status.

REQUEST CFILE.

This format requests the operator at the system display console to assign a specific peripheral equipment to this job with the name CFILE. This must be the first appearance of the name CFILE. The job waits for operator action before proceeding.

MODE 3.

This format assigns arithmetic exit mode 3 to the job until further specification.

SWITCH 5.

This format sets switch 5 for reference by subsequent FORTRAN programs.

EXIT.

This format is used to separate the control cards associated with the normal execution of the job from a group of control cards to be executed in the event of an error exit. If this control statement is reached by sequential execution of the job it terminates the job. If an error exit occurs in the job before this EXIT card is reached, the control statements following this card are executed.

Sample Job -

The following job cards are an example of a Fortran load and run job with READ and PRINT statements. The job has a priority of 2, a time limit of one minute, and a field length of 40000 octal words. A Fortran program deck may be declared in either a FORTRAN II or FORTRAN IV mode. The mode is declared in the header card for the main program along with the name and I/O file arguments.

```
JOB236, 2, 100, 40000.
RUN.
789
FORTRAN IV PROGRAM JOHN(INPUT,OUTPUT)

(Fortran program coding)

789

(Fortran data)

6789
```

Sample job -

The following job illustrates a Fortran load and run job with three tape references. The Fortran language TAPE 1 in this example is assumed to be an input tape which the operator loads on a particular unit. The other two tapes are assumed to be scratch tapes which are draw from a tape pool.

```
JOB237, 2, 400, 60000.
ASSIGN WT, TAPE 5.
ASSIGN WT. TAPE 6.
REQUEST TAPE 1.
RUN.
789
FORTRAN II PROGRAM HENRY(INPUT,OUTPUT, TAPE1,TAPE5,TAPE6)

(Fortran program coding)

789

(Fortran data)

6789
```

Monitor Action on Central Programs -

The monitor will act upon a central program in the following cases.

Higher Priority.

If a higher priority program is ready for execution the current program exchanges in favor of the new program.

Arithmetic Exit.

The monitor exchanges to the next lower priority program whenever the central program address becomes zero. An arithmetic exit mode flag is set for this case.

Time Limit.

The monitor exchanges to the next lower priority program if the time limit is reached. A time limit flag is set in this case.

Call Peripheral.

A central program calls a peripheral package by entering the package name in display code in the upper 18 bits of RA + 1. The lower 36 bits may contain arguments for the peripheral package. The monitor clears address RA + 1 as soon as the peripheral package is called.

Recall.

A central program may stop for later recall by entering RCL in RA + 1 and looping until RA + 1 clears. The monitor will exchange to the next lower priority program and restart this program at a later time (about 250 milliseconds). If the central program has called a peripheral package for I/O action this package may restart the central program when the action is completed.

Abort.

A central program may abort the job by entering ABT in RA + 1. This action sets an abort exit flag.

End.

A program is terminated by entering END in RA + 1. The monitor then calls a peripheral package to advance the job to the next control statement.

Central Program Formats -

Binary versions of central programs contain data in the forepart of the program code which controls loading and starting of the program. This data is contained in the first two words of the program code. These words are interpreted by the loading processor and then cleared before beginning execution.

Central program arguments begin in address 000002 of a central program. The arguments are file names which are in display code left adjusted in the word. A set of file names are generally stored in the argument area of a program at the time it is compiled. If arguments are used in a control card at the time the program is called for execution, these arguments are inserted in the program in the argument area and override the original values. If no arguments are used, or if fewer than anticipated arguments are used, the original argument values are effective at run time. The list of argument names at the front of the program must be followed by a cleared word. The first word of program for execution follows this cleared word.

The first word of a binary central program contains the following information.

(42 bits) Program name in display code.
(18 bits) Program length.

The second word of a binary central program contains the following information.

(54 bits) Not used.
(6 bits) Number of arguments.

Execution of a central program begins at an address equal to the number of arguments plus 3.

Resident Subroutine Library -

The following central program subroutines are stored in the resident subroutine library.

ALOG Computes the natural logarithm of a floating point number. The result for a negative argument is indefinite.

ALOG10 Computes the common logarithm of a floating point number. The result for a negative argument is indefinite.

ATAN Computes the arctangent of a floating point number.

COS Calculates the cosine of a floating point number.

DVCHK Divide and check.

END Causes end of file on all circular buffers for the program.

EXIT Terminates a program similar to END. The word EXIT is entered in the dayfile.

EXP Calculates the floating point function E * * X.

IBAIEX Computes I * * J where both I and J are fixed point.

IFENDF Checks the status of a circular buffer for an end of file condition. IN = OUT and 3X function.

OVERFL Result is set to one if overflow exits and two if no overflow.

PAUSE Loop on recall until operator action. Word PAUSE entered in the dayfile.

RBAIEX Computes A * * I where A is floating point and I is fixed point.

RBAREX Computes A * * B where both A and B are floating point.

REMARK Transmits a message to the system dayfile and displays it on the console display.

Example: CALL REMARK (22H REMARKS HERE IN ENTRY).
22H indicates 22 Hollerith characters in the remarks (maximum of 40).

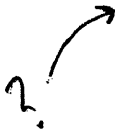
SIN Computes the sine of the argument.

SQRT Computes the square root of the argument. A negative argument results in an indefinite result.

SLITE Sense lights. Turn off if zero. Turn on if non zero.

SLITET Respond with one if light was on, two if light was off. Turn off light.

SSWTCH	Respond with one if switch is down, two if switch is up.
START	Enter START in dayfile.
STOP	Enter STOP in dayfile and perform same function as END.
TAN	Compute the tangent of the argument.
TANH	Compute the hyperbolic tangent of the argument.
TIME	Enter the word TIME and hollerith information in the dayfile.



Library Subroutine Calling Sequences

Arithmetic subroutines in the Resident Subroutine Library may be called by the following calling sequences.

I.	ALOG	COS	SQRT
	ALOG10	SIN	TAN
	ATAN	EXP	TANH

Each of the above subroutines is entered with the address of its single argument in the B1 register and exits with its result in the X6 register. All registers except A0 are available and need not be saved.

Example
AA = SIN(BB)

Program Address			
CSEQ	I = BB	6110B	Put address of argument in B1
	R = SIN	0100 SIN	Return Jump to SIN subroutine
	Y = (AA)	506..	Store answer from X6 to memory
	.		
	.		
	.		
	Continuation of main program		
SIN-3			Subroutine name and length
SIN-2			Number of arguments
SIN-1			
SIN	0	00...	Entry/Exit Address
	.		
	.		
	.		
	P = SIN	02..SIN	Jump to Entry/Exit Address of Subroutine for return to main program

II. The subroutines IBAIEX, RBAIEX, and RBAREX require two arguments. X registers 6 and 7 are used for transmitting the arguments, X6 for the base and X7 for the exponent. These subroutines save and restore all index registers used. The result is left in X6.

Example
AA = BB**CC

Program Address			
CSEQ	Y = BB	6170 BB	Set X6 to desired base
	Z = CC	6160 CC	Set X7 to desired exponent
	R = RBAREX	0100 RBAREX	Return Jump to RBAREX subroutine
	G = AA	5160 AA	Store result at AA
	.		
	.		
	.		
	Continuation of main program		
RBAREX	0	00	Entry/Exit
	.		
	.		
	.		
	P = RBAREX	02...	Jump to Entry/Exit address for return to main program

III. The BACKSP, REWINM, ENDFIL, and IFENDF subroutines receive a single argument via the X6 register and do not use index registers. The calling sequence must place in X6 either:

A. The address of the parameter list describing the circular buffer to be referenced, or

B. The complement of the address of the file name or variable tape number. File names are left justified and tape numbers are binary and right justified.

If the address of the parameter list is known at the time the calling program is compiled, the calling sequence may place this address in X6 before jumping to the subroutine.

If the address of the parameter list is not known at compile time, the calling sequence must place in X6 the complement of the address of the tape number or file name. When the subroutine senses that X6 holds a complemented address (<0), it determines whether the word at that address is a file name or tape number and, if necessary, forms a file name by inserting TAPE in display code in front of the tape number (also converted

to display code). When the proper file name has been obtained, a masked search is made starting at address 000002 of the program. The address of the parameter list is found in the lower 18 bits of the match.

Example
REWINM 6

When address of parameter list is known:

Program Address			
CSEQ	Y = PLIST	7160 PLIST	Set X6 to address of parameter list
	R = REWINM	01...	Return Jump to REWINM subroutine
	.		
	.		
	.		
	Continuation of main program		
REWINM	0	00	Entry/Exit
	.		
	.		
	.		
	P = REWINM	0200 REWINM	Jump to Entry/Exit address for return to main program
PLIST	TAPE6...SS		File name in display code and buffer status
	FIRST		First address of buffer
	IN		
	OUT		
	LIMIT		Last address of buffer

When address of parameter list is unknown (variable tape number):

```
Program
Address

CSEQ        Y = TAPENO          7160 TAPENO        Set X6 to address of
                                                   tape number

            Y = -Y              14606              Complement address in X6

            R = REWINM          01                 Return Jump to REWINM
                                                   subroutine
            .
            .
            .
            Continuation of main program


REWINM      0                   00                 Entry/Exit

            .
            .
            .
            P = REWINM          02 REWINM          Jump to Entry/Exit
                                                   address for return to
                                                   main program

TAPENO      0000....6                              Tape number (binary)


RA + 2      File Name 1
RA + 3      File Name 2

.
.

RA + n      TAPE 6 .. PLIST                        Matching file name and
                                                   address of parameter list
```

When address of parameter list is unknown (File name):

```
CSEQ        Y = FILENAME        7160 FILENAME      Set X6 to address of
                                                   file name

            Y = -Y              14606              Complement address in X6

            R = REWINM          01...REWINM        Return Jump to REWINM
                                                   subroutine


FILE NAME   TAPE6000..0
```

IV. The subroutines INPUTB, INPUTC, OUTPTB, AND OUTPTC assume that all registers are available. These subroutines receive arguments via the B registers and are entered at least three times for each data list transferred:

On first entry B1 = 000000, B2 contains the address of the parameter list or the complement of the address of the file name or tape number, and B3 contains the address of the format statement (for the coded routines only).

On intermediate entries B1 contains the address of a data item or the beginning address of an array, and B2 is the array length.

On final entry B1 < 0 indicating that the data list has been exhausted.

Example

List the data in array AA and variable BB.

Program Address			
CSEQ	I = 0	61100	Set B1 = 000000
	J = PLIST	61200 PLIST	Set B2 to address of parameter list
	K = FORMAT	61300 FORMAT	Set B3 to address of format statement
	R=OUTPTC	0100 OUTPTC	Return Jump to OUTPTC subroutine
	I = A	61100 AA	Set B1 to beginning address of array AA
	J = LENGTH	61200 LENGTH	Set B2 to length of array AA
	R = OUTPTC	0100 OUTPTC	Return Jump to OUTPTC subroutine
	I = BB	6110 B	Set B1 to address of variable BB
	J = 0	61200 0	Set B2 to 000000
	R = OUTPTC	0100 OUTPTC	Return Jump to OUTPTC
	I = -1	6110 -1	Set B1 < 0 to indicate all data has been transferred

```
R = OUTPTC          0100 OUTPTC        Return Jump to OUTPTC
                                       subroutine to send E.O.R.
      .
      .
      .
Continuation of main program
```

Central Library Fortran Subroutines -

XLOCF Returns the memory location of a variable name.

INPUTB Binary input.

INPUTC Fortran data input.

OUTPTB Binary output.

OUTPTC Fortran data output.

ENDFIL Write end of file.

REWINM Rewind medium.

BACKSP Backspace medium.

DISPLA Displays a variable name and its numerical value. The value is displayed as an integer if unnormalized and in floating point format if normalized.

Example: CALL DISPLA (5HVNAME, VNAME)
Displays a variable with 5 Hollerith characters (VNAME) and its numerical value.

RANF Random number generator.

CHAIN Loads a program from the disk file and executes the program. Arguments transferred from one program to another must be in the common region. All segments to be chained must be compiled with the same file names. The segment name is transferred to the dayfile and displayed prior to execution.

Example: CALL CHAIN (5HLINK2) calls and executes program LINK2. 5H indicates 5 Hollerith characters in the name.

Central Library Programs -

RUN - Compile and run fortran.

COPY - Copy to double file mark.

COPYCR - Copy coded record.

COPYCF - Copy coded file.

COPYBR - Copy binary record.

COPYBF - Copy binary file.

REWIND - Rewind medium.

VERIFY - Verify two media.

APRAB - Assemble peripheral overlay.

COPYSBF - Copy shifted binary file.

COPYCR(A,B,N) - Copy Coded Record

This central program copies N coded records from a medium A to a medium B. If the third argument is omitted one record is copied.

COPYCF(A,B,N) - Copy Coded File.

This central program copies N coded files from a medium A to a medium B. If the third argument is omitted one file is copied.

COPYBR(A,B,N) - Copy Binary Record.

This central program copies N binary records from a medium A to a medium B. If the third argument is omitted one record is copied.

COPYBF(A,B,N) - Copy Binary File.

This central program copies N binary files from a medium A to a medium B. If the third argument is omitted one file is copied.

COPYSBF(A,B) - Copy Shifted Binary File.

This central program copies a binary file of coded information from a medium A to a medium B, shifting each line one character and adding a leading space.

REWIND(A) - Rewind Medium.

This central program rewinds a tape or disk medium.

BKSP(A) - Backspace Medium.

This central program backspaces a tape or disk medium.

VERIFY(A,B) - Verify Two Media.

This central program reads two files and compares the data word by word to a file mark. If a disagreement is found the program stops. The word from the first file is contained in X1 and the word from the second file in X2. The program may be resumed by advancing the program address and restarting.

COPY(A,B) - Copy to double file mark.

This central program copies from one medium to another until a double file mark is detected. Both media are then backspaced over the last file mark.

FORTRAN COMPILER ARGUMENTS

The Fortran compiler, RUN, may be called with up to six arguments. An example appears below, and the arguments are identified in the order of appearance:

RUN(L, 300000, 100000, 3000, AA, BB)

(1) Compiler mode.
    G - Compile and execute.
    S - Compile and do not execute.
    P - Compile and punch and do not execute.
    L - Compile and list and do not execute.
(2) Object program field length. (octal)
(3) Object program common length. (octal)
(4) Object program I/O buffer lengths. (octal)
(5) File name for compiler input.
(6) File name for compiler listable output.

If an argument is entered as zero, or a space, or the list of arguments is shorter than six, an assumed argument is entered by the compiler. Compiler output, except in the G mode, includes a reproduction of the source program, a variable map, and indications of errors detected during compilation. If the G mode is selected, all output is suppressed unless errors are detected, in which case the output is the same as indicated for the other modes. If the L mode is selected, the output will include an octal list of the compiled instructions.

(1) If the first argument is omitted it is assumed to be G.

(2) If the second argument is omitted it is set equal to the field length at compile time.

(3) If the third argument is omitted it is set equal to the amount of common storage required for the main program being compiled.

(4) The fourth argument specifies the length of the buffers which will be used for each I/O device in the object program. If not specified this argument is assumed to be 2001 octal.

(5) If this argument is omitted it is assumed to be INPUT.

(6) If this argument is omitted it is assumed to be OUTPUT.

A copy of the compiled program is always left in disk storage as a binary file with the name of the program as file name. It may be called and executed repeatedly by name.

FORTRAN PROGRAM HEADER CARDS

Every Fortran program must have a header card in one of the following forms:

```
PROGRAM NAME (A1, ... , AN)
FORTRANIV PROGRAM NAME (A1, ... , AN)
FORTRANII PROGRAM NAME (A1, ... , AN)
```

The first two forms cause the program and its subroutines to be compiled in the FORTRAN IV mode. The third form causes compiling in the FORTRAN II mode.

The arguments A1, ... , AN must be the names of all input/output files required by the main program and its subroutines. Although these arguments may be changed at the time the program is executed, they must, at compile time, satisfy the following conditions:

1. The file name INPUT must appear if any READ statement is included in the program or its subroutines.

2. The file name OUTPUT must appear if any PRINT statement is included in the program or its subroutines.

3. The file name PUNCH must appear if any PUNCH statement is included in the program or its subroutines.

4. The file name TAPEi, where i is an integer, must appear if a READ INPUT TAPE i, WRITE OUTPUT TAPE i, READ TAPE i, WRITE TAPE i, READ (i,n), WRITE (i,n), READ (i) or WRITE (i) statement is included in the program or its subroutines.

5. The file names $TAPEi_1$, ..., $TAPEi_k$ must appear if I is an integer variable name and a READ INPUT TAPE I, WRITE OUTPUT TAPE I, READ TAPE I, WRITE TAPE I, READ (I,n), WRITE (I,n), READ (I) or WRITE (I) statement appears in the program or its subroutines. The integers i_1, ..., i_k must include all values which will be assumed by the variable I. The file name TAPEI may not appear in the list of arguments to the main program.

6. The file name TAPEV must appear if V is a symbolic tape unit number and a READ INPUT TAPE V, WRITE OUTPUT TAPE V, READ TAPE V, WRITE TAPE V, READ (V,n), WRITE (V,n), READ (V) or WRITE (V) statement appears in the program or its subroutines

A program is terminated by a single END card.

A provision is made for equivalencing file names at compile time. For example, the header record

```
PROGRAM NAME (INPUT, OUTPUT, TAPE1 = INPUT, TAPE2 = OUTPUT)
```

would cause all input normally provided by TAPE1 to be extracted from the INPUT file and all listable output normally recorded on TAPE2 to be transmitted to the OUTPUT file. Equivalenced file names must follow, in the list of arguments, those to which they are made equivalent. Their corresponding argument positions may not be changed at the time the program is executed, although the names of the files to which they are made equivalent may be changed at this time.

SUBROUTINE HEADER CARDS

Every Fortran SUBROUTINE compiled independently of a main program which is to use it must have a header card in one of the following forms:

```
SUBROUTINE NAME (A1, ..., AN)
FORTRANIV SUBROUTINE NAME (A1, ..., AN)
FORTRANII SUBROUTINE NAME (A1, ..., AN)
```

The first two forms cause the subroutine to be compiled in the FORTRAN IV mode, and the third form causes it to be compiled in the FORTRAN II mode. The arguments A1, ..., AN have their conventional FORTRAN significance.

A SUBROUTINE is terminated by a single END card.

FUNCTION HEADER CARDS

Every Fortran FUNCTION which is compiled independently of a main program which is to use it must have a header card in one of the following forms:

```
FUNCTION NAME (A1, ..., AN)
FORTRANIV FUNCTION NAME (A1, ..., AN)
FORTRANII FUNCTION NAME (A1, ..., AN)
```

The first two forms cause the function to be compiled in the FORTRAN IV mode, and the third form causes it to be compiled in the FORTRAN II mode. The arguments A1, ..., AN have their conventional FORTRAN significance.

A FUNCTION is terminated by a single END card.

FORTRAN HANDLING OF BINARY SUBROUTINES

The Fortran compiler, RUN, is capable of processing previously compiled or assembled subroutines which appear on binary cards. If this capability is to be used, then a card with a plus sign (+) in column 1 must immediately follow the END card of the program to be compiled. This card, in turn, must be immediately followed by subroutines on binary cards. The end of the binary subroutine portion of the deck is indicated by a standard end-of-file card.

```
JOB238(5,100,40000)
RUN
789
PROGRAM HENRY (INPUT, OUTPUT)

(Standard Fortran coding)

END
+

(Binary subroutines)

789

(Fortran data)

6789
```

ADDITIONAL NOTES ON THE FORTRAN COMPILER

The following remarks concern certain nonstandard features of the compiler;

1. Boolean constants in B-type statements must be changed in some cases because of the increased word length of the 6600 or because of the use of display code as an internal code.

2. Complex arithmetic is not performed by subroutines.

3. Double-precision arithmetic is performed on one-word operands (allowing approximately 15 decimal digits of accuracy), although two memory locations are allocated to double-precision variables.

4. The notations ".T." and ".F." may be substituted for the FORTRAN IV constants ".TRUE." and ".FALSE.".

5. The compiler requires 30000 octal locations in memory plus a variable-sized area for tables and the binary version of the object program. It is recommended that arrays be assigned to the blank COMMON region since this region is not part of the binary version of the object program as developed at compile time.

6. Independently compiled subroutines, or functions, may not use more than 65,536 decimal locations in the COMMON region or more than 65,536 decimal locations locally.

7. FORTRAN IV blank COMMON assignments are made after labeled COMMON assignments, i.e., blank COMMON regions are located in higher addresses than labeled COMMON regions. A labeled COMMON region may be considered as a local region of the program or subroutine which establishes it. Hence, data may be entered into it by DATA statements.

8. Object-program input/output buffers are assigned to the top of the field in which the program is to run. Blank COMMON assignments are made to the region immediately below the input/output buffers. Neither of these regions is included in the binary version of a program or subroutine which is transmitted to the disk for running or punching.

9. FORTRAN II and FORTRAN IV statements which are not inherently incompatible may be intermixed in a program to be compiled in either mode. Inherently incompatible statements are those involving function subroutine references and EQUIVALENCE statements causing a reordering of variables in the COMMON region. However, any standard FORTRAN II or FORTRAN IV library or built-in subroutine reference may appear in a program to be compiled in either mode.

10. The statement IF(ENDFILE I)N1,N2, where I is a constant or variable, may be used in a program or subroutine.

11. If a program is declared to be FORTRAN II, FORTRAN IV, MACHINE, or ASCENTF, this mode is assumed for all subsequent subroutines unless specific subroutines are declared to be of a different mode. If a subroutine is declared to be of a different mode, it will be processed in its declared mode. The subroutine following it, unless again declared to be of a different mode, will be processed in the mode applying to the main program. In other words, after processing any subroutine the compiling mode reverts to the one applying to the main program.

12. Indices in DO loops must be less than 2^{17} in absolute value.

13. A BLOCK DATA subroutine is not necessary for entering data into a labeled COMMON region, although it may be used if desired.

14. An 0 followed by at least six octal digits is interpreted as an octal number by the compiler and is assigned a logical mode.

15. A two-way IF statement is allowed. For example, IF(E)N1,N2 causes branching to statement N2 only if the value of E is +0. E may be of logical, integer, real, double, or complex mode. If its mode is complex, the test is made on its real part.

16. Full mixed mode arithmetic is permitted. The mode of an arithmetic expression is determined by the term of highest mode appearing in the expression, where the modes, from highest to lowest, are COMPLEX, DOUBLE, REAL, INTEGER, and LOGICAL. An expression is of logical mode if .NOT., .OR., or .AND. appears as an operator or if it contains a single term of logical mode. In this case, operands may be of any mode. If an operand is complex, a logical operation is performed on its real part.

17. If L is a logical variable and E is an expression of any mode, L = E causes a replacement of L by the value of E.

18. If V is a variable of any mode and L is a logical expression, V = L causes a replacement of V by the value of L.

19. A slash (/) in a B-type FORTRAN II statement causes a logical difference to be formed on its associated operands.

20. If C is a complex variable and E is an expression of any mode except complex, C = E causes a replacement of the real part of C by the value of E and a replacement of the imaginary part of C by +0.

21. If V is a variable of any mode except complex and E is a complex expression, V = E causes a replacement of V by the real part of E.

22. The logical operators .NOT., .OR., or .AND. may be replaced by .N., .O., or .A., respectively.

23. A number with fewer than six digits trailed by an S and appearing as an argument to a subroutine call is interpreted by the compiler as a statement number; e.g., CALL DUMP(100S,200S,0) causes an octal dump of the region between statements 100 and 200 of the program being compiled.

24. All working regions of object programs are clear prior to execution, except for blank COMMON regions. Blank COMMON regions are clear at execution time if compilation is in the G mode.

FORTRAN HANDLING OF ASSEMBLY LANGUAGE

The Fortran compiler, RUN, is capable of processing programs or subroutines written in assembly language. Such programs or subroutines may be intermixed with regular Fortran programs and subroutines. Each must be organized as follows:

1. Header card.
2. Fortran cards, if any.
3. Declaration cards, if any.
4. Instruction cards.
5. Constant cards.
6. End card.

It is noted that

a) The instruction portion of the deck must be preceded by "0" lines corresponding to control words, arguments, and an exit/entry line.

b) The constant portion of the deck must be separated from the instruction portion by a card with two periods (..) punched in columns 7 and 8.

c) The end card is punched as in Fortran, i.e., END in columns 7-9 of a card.

d) Constants may appear in the instruction portion of the deck provided they are positive and less than 2^{54}.

e) A card with an asterisk (*) in column 1 may appear anywhere in the deck and is treated as a remark card.

f) A card with a period (.) in column 1 may appear anywhere in the deck and will cause a page eject at the time the program or subroutine is listed.

Header Formats:

Each program or subroutine coded in assembly language must have a header card in one of the following formats:

```
MACHINE PROGRAM NAME
MACHINE PROGRAM NAME (A1, ..., AN)
MACHINE SUBROUTINE NAME
MACHINE SUBROUTINE NAME (A1, ..., AN)
```

It is noted that

a) The header information must be punched between column 6 and column 73 of each card used.

b) Up to 19 continuation cards may be used in any declaration, but an asterisk (*) must appear in column 6 of each continuation card.

c) In the forepart of the program or subroutine to be assembled there must be three "0" lines plus one "0" line for each argument A1, ..., AN:

```
        0       CONTROL WORD ONE
        0       CONTROL WORD TWO
A1      0       ARGUMENT 1
:       :
.       .
AN      0       ARGUMENT N
        0       EXIT/ENTRY
```

The first two "0" lines correspond to control information furnished by the compiler, and the last "0" line is unused by a program but is the exit/entry line for a subroutine. The first executable instruction must follow the exit/entry line.

d) The arguments A1, ..., AN are treated as dummy arguments by the compiler in that they are used only to obtain an argument count to insert in the second control word.

e) If an assembly-language subroutine is to be referenced by a Fortran program or subroutine, then the assembly-language subroutine must be written assuming that the addresses of the first six arguments, A1-A6, will be transmitted through index registers B1-B6; addresses of arguments beyond the sixth, A7-AN, will be transmitted into the locations corresponding to A7-AN within the assembly-language subroutines; a return jump will be made to the location following AN.

f) Function routines, either Fortran-coded, assembly-language coded, or from the system's library, leave results in X6 upon exiting.

Fortran Formats:

Fortran statements which are allowed in an assembly-language program or subroutine are the following:

COMMON
EQUIVALENCE
DIMENSION
EXTERNAL
DATA

Identifiers appearing in the above statements may be used in subsequent symbolic instructions. Continuation cards must contain an asterisk (*) in column 6.

Declaration Formats:

Six tag-associating declarations are allowed in assembly language. These provide for associating alphanumeric tags with constants, with regions reserved locally or in COMMON, and with external subroutines which are subject to reference. Examples and explanations of these declarations follow:

1. CON (C1=25, C2=777B, C3=-6.54E-2)

   Causes the constants on the right of the equals relations to be assembled into a storage area and tagged with the identifiers appearing on the left.

2. HOL (H1=ABCDEFGHIJ, H2=1234567890)

   Causes the ten-character groups, including spaces, on the right of the equals relations to be converted to display code, placed into a storage area, and tagged with the identifiers appearing on the left.

3. ABS (JJ=100, KK=100B, LL=7777B)

   Causes the unsigned values on the right of the equals relations to be assembled into instructions containing the tags on the left in their address fields.

4. RES (K1=10, K2=100B, K3=1000)

   Causes local block reservations, where the number of words reserved in each block is the unsigned number to the right of the equals relation and where the beginning of each block is tagged with the identifier on the left.

5. COM (B1=1, B2=300, B3=205B)

   Causes blank COMMON block reservations, where the number of words reserved in each block is the unsigned number to the right of the equals relation andwhere the beginning of each block is tagged with the identifier on the left.

6. SUB (SI=SIN, LG=LOG, OUT=OUTPTC)

   Causes the subroutines whose names appear on the right of the equals relations to be assembled into memory and tagged with the identifiers appearing on the left.

It is noted that

a) Each tag-associating declaration is punched between column 6 and column 73 of each card used.

b) Up to 19 continuation cards may be used in any declaration, but an asterisk (*) must appear in column 6 of each continuation card.

c) The open parenthesis (( ) following CON, HOL, ABS, RES, COM, or SUB may be replaced by any separator if the final closing parenthesis ( )) is dropped.

Instruction Formats:

In the assembly language, operational registers are designated by single-character names as follows:

S = X0	0 = B0	A = A0
T = X1	I = B1	B = A1
U = X2	J = B2	C = A2
V = X3	K = B3	D = A3
W = X4	L = B4	E = A4
X = X5	M = B5	F = A5
Y = X6	N = B6	G = A6
Z = X7	Ø = B7	H = A7

The letter R is used to specify a return jump, and the letter P is used to specify all other jumps.

Let S represent any of the letters S-Z, I represent any of the letters I-O or the digit 0, and A represent any of the letters A-H. Let Q represent a positive integer less than 2^{16} or an alphanumeric tag of 2-6 characters. Then the forms of assembly-language instructions, grouped according to functional units required for execution, are as follows:

Symbolic Form	Machine Form	Example
0	00xxx	0
R=Q	01xxK	R=TAG
P=Q+I	02ixK	P=TAG+N
P=Q,S=0	030jK	P=TAG,T=0
P=Q,S/0	031jK	P=TAG,U/0
P=Q,S)0	032jK	P=TAG,V)0
P=Q,S(0	033jK	P=TAG,W(0
P=Q,S.I	034jK	P=TAG,X.I
P=Q,S.Ø	035jK	P=TAG,Y.Ø
P=Q,S.D	036jK	P=TAG,Z.D
P=Q,S.N	037jK	P=TAG,S.N
P=Q,I=I	04ijK	P=TAG,J=K
P=Q,I/I	05ijK	P=TAG,L/M
P=Q,I)I	06ijK	P=TAG,N)Ø
P=Q,I(I	07ijK	P=TAG,I(0
S=S	10ijx	Y=V
S.L=S*S	11ijk	T.L=W*X
S.L=S+S	12ijk	V.L=Y+Z
S.L=S-S	13ijk	V.L=U-V
S=-S	14ijk	Z=-W
S.C=S*S	15ijk	W.C=T*U
S.C=S+S	16ijk	X.C=V+W
S.C=S-S	17ijk	Y.C=S-Z

Symbolic Form	Machine Form	Example
S=S(Q)	20ijk	T=T(24)
S=S(-Q)	21ijk	U=U(-10)
S=S(I)	22ijk	V=X(N)
S=S(-I)	23ijk	W=W(-M)
S,I=S-	24ijk	X,J=W-
S,I=S+	25ijk	Y,K=Z+
S,I=S.	26ijk	Z,0=U.
S=I,S.	27ijk	T=K,V.
S=*Q	43ijk	Y=*12
S.N=S+S	30ijk	T.N=U+V
S.N=S-S	31ijk	U.N=X-Z
S.D=S+S	32ijk	V.D=V+V
S.D=S-S	33ijk	W.D=S-U
S.R=S+S	34ijk	X.R=Y+Z
S.R=S-S	35ijk	Y.R=Z-T
S.I=S+S	36ijk	T.I=U+X
S.I=S-S	37ijk	Z.I=Y-Z
S.N=S*S	40ijk	S.N=X*X
S.R=S*S	41ijk	Y.R=Y*Y
S.D=S*S	42ijk	T.D=T*U
S.N=S/S	44ijk	Y.N=T/X
S.R=S/S	45ijk	Z.R=X/W
$	46xxx	$
S=*S	47ixk	T=*W
S=(A+Q)	50ijK	T=(C+TAG)
S=(I+Q)	51ijK	U=(J+100)
S=(S+Q)	52ijK	Z=(T+30B)
S=(S+I)	53ijk	T=(T+J)
S=(A+I)	54ijk	U=(B+K)
S=(A-I)	55ijk	V=(C-N)
S=(I+I)	56ijk	W=(M+N)
S=(I-I)	57ijk	X=(L-K)
I=A+Q	60ijK	J=H+TAG
I=I+Q	61ijK	K=L+10
I=S+Q	62ijK	L=L+55B
I=S+I	63ijk	M=T+J
I=A+I	64ijk	N=G+K
I=A-I	65ijk	Ø=A-L
I=I+I	66ijk	I=I+J
I=I-I	67ijk	J=K-M
S=A+Q	70ijK	T=G+TAG
S=I+Q	71ijK	U=K+5
S=S+Q	72ijK	V=L+15B
S=S+I	73ijk	W=X+J
S=A+I	74ijk	X=A+K
S=A-I	75ijk	Y=C-L
S=I+I	76ijk	Z=M+I
S=I-I	77ijk	S=N-Ø

It is noted that

a) The arithmetic mode indicators L, C, N, D, R, and I may immediately follow a result register name, i.e., the period (.) in these cases is optional.

```
TL=X*Y
UC=V+V
VN=T/W
WD=X+Y
XR=S-T
```

b) In the instructions 02 and 50-77 either term may be dropped, in which case a 0 designation is assembled.

```
P=K
P=TAG
T=(J)
J=15B
U=-K
```

c) In the instructions 50-54, 60-64, and 70-74 the terms may be interchanged unless Q is a constant.

```
T=(TAG+L)
M=K+B
U=L+X
```

d) In the instructions 50-52, 60-62, and 70-72 the plus sign (+) may be replaced by a minus sign (-) if Q is a constant.

```
X=(I-30)
J=K-55B
Y=-1
```

e) In the instructions 51, 61, and 71 the right member may be an indicated sum or difference of a tag and a constant, in which case the constant must follow the tag.

```
W=(TAG-35)
K=TAG+1
U=TAG+100B
```

f) In the instruction 51 the parenthesized quantity may be a constant, represented in conventional Fortran form, only if the result register is to receive the machine version of that constant; in this case the address of the converted number is assembled into the instruction.

```
T=(-1.5E-6)
U=(47550516045547B)
   *BEND
```

g) If it is desired to have Q correspond to an octal integer, then the digits in the number must be trailed by a B.

h) Alternate forms for certain instructions are

S=S.S	11ijk
S=S$S	12ijk
S=-S.S	15ijk
S=-S$S	16ijk
S=S+S	36ijk
S=S-S	37ijk
S=S*S	40ijk
S=S/S	44ijk
A=A+Q	50ijK
A=I+Q	51ijK
A=S+Q	52ijK
A=S+I	53ijk
A=A+I	54ijk
A=A-I	55ijk
A=I+I	56ijk
A=I-I	57ijk

i) Each instruction must be punched between column 6 and column 73 of a card; no blanks are permittedwithin the instruction code.

j) A comment may follow any instruction code, but at least one blank must separate it from the instruction.

k) An alphanumeric location tag of 2-6 characters may be punched in columns 1-6 of a card containing an instruction; no blanks are permitted within the tag.

l) A plus sign (+) in a location field will force the corresponding instruction to the high order positions of a new word.

m) The instruction 00 is assembled as a full zero word.

Constant Formats:

Decimal constants in standard Fortran notation may be specified in assembly-language. Octal constants may be specified by placing a B after the digits of the number.

It is noted that

a) A constant must be punched between column 6 and column 73 of a card; no blanks are permitted with the constant specification.

```
-100.0
500
100B
+15.64E-3
```

b) An alphanumeric location tag of 2-6 characters may be punched in columns 1-6 of a card containing a constant; no blanks are permitted within the tag.

```
CON1    -150.0
CON2    3.6E10
```

c) Block reservations of zero words, tagged or untagged, may be made by enclosing the number of words to be reserved in parentheses; the parenthesized quantity must appear between column 6 and column 73 of a card; if a location tag appears on the same card, then it will be associated with the first word of the block.

```
BK1     (100)
BK2     (200B)
```

FORTRAN HANDLING OF AN ASCENT SUBSET

The Fortran compiler, RUN, is capable of processing programs or subroutines written in a subset of ASCENT assembly language. Such programs or subroutines may be intermixed with regular Fortran programs and subroutines. Each must be organized as follows:

1. Header card.
2. Fortran cards, if any.
3. Instruction cards.
4. Constant cards.
5. End card.

It is noted that

a) The instruction portion of the deck must be preceded by lines of coding which produce "0" words corresponding to control words, argument words, and an exit/entry word.

b) The instruction portion of the deck may contain BSS, BSSZ, and EQU cards. The address field of any such card may contain only a single constant. BSS and BSSZ cards produce zero regions.

c) The constant portion of the deck may contain BSS, BSSZ, EQU, DPC, BCD, and CON cards. The address fields of these cards may contain only a single constant or ten-character string of the form *ABCDEFGHIJ*. DPC and BCD character strings are reduced to display code.

d) The constant portion of the deck must be separated from the instruction portion by a card with two periods (..) punched in columns 7 and 8.

e) The end card is punched as in Fortran, i.e., END appears between column 6 and 73 of the card.

f) Fortran C-type comment cards are not permitted.

g) Fortran cards may contain COMMON, EQUIVALENCE, DIMENSION, EXTERNAL, or DATA statements. Identifiers appearing in the statements may be used in subsequent symbolic instructions. Continuation cards must contain an asterisk (*) in column 6.

h) The header card must be in one of the following formats:

```
ASCENTF PROGRAM NAME
ASCENTF PROGRAM NAME (A1,...,AN)
ASCENTF SUBROUTINE NAME
ASCENTF SUBROUTINE NAME (A1,...,AN)
```

The header information must be punched between column 6 and column 73 of each card used. Continuation cards may be used, but must be designated by an asterisk (*) in column 6.

i) Instruction formats are as described in ASCENT programming manuals, but with certain restrictions. An address field may contain an indicated sum of a tag and constant, but not a sum or difference of two tags. Location tags may start in column 1, but may not extend beyond column 6. Instructions may start anywhere beyond column 6, but no card may contain more than one instruction. The PS instruction causes assembly of a full zero word.

j) Location tags associated with pseudo-operations may start in column 1, but may not extend beyond column 6.

k) Double-precision and complex "literal constants" are not accepted.

l) A minus sign (-) in a location field is not allowed.

m) An asterisk(*) in an address field is not allowed.

n) Except for the header card and the double-period card, separating instructions from constants, upward compatibility from the Chippewa System to SIPROS is achievable by starting location tags of no more than five characters in column 2 and by starting instructions and pseudo-operations in column 11.

o) All other formats, except those of instructions, which have been described for the Chippewa assembly language acceptable to the compiler may be used; but each such usage introduces an incompatibility between the Chippewa System and SIPROS. If one desires only to replace instruction formats of the Chippewa assembly language by ASCENT formats, but retain all other formats of that language, he may do so.

FORTRAN ERROR PRINTOUTS

During a Fortran compilation two-character error prints may follow statements which are incorrect; other such prints may follow the END statement, indicating other types of errors in the program. A list of these two-character error indicators, along with a brief explanation of the type of error which each indicates, follows:

AC Argument-Count Error

Indicates that the number of arguments in a current reference to a subroutine differs from the number which occurred in a prior reference.

AL Argument-List Error

Indicates a format error in a list of arguments.

AS Assign Error

Indicates a format error in an ASSIGN statement.

BC Boolean-Constant Error

Indicates a format error in the designation of a FORTRAN boolean constant in a B-type expression.

BI Binary-Input Error

Indicates that a subroutine on binary cards following the Fortran program and its Fortran-coded subroutines has an incorrect header card.

BO Common-Block-Overflow Error

Indicates that the current requirements for a labeled block of COMMON storage exceed the length of the block as established in a preceding COMMON statement.

BX Boolean-Expression Error

Indicates a format error in a B-type boolean statement.

CD Duplicate-Common Error

Indicates that a variable currently being assigned to the COMMON region has been previously assigned to this region.

CE Common-Equivalence Error

Indicates that two variables assigned to COMMON storage are being improperly equivalenced.

CL Call Error

Indicates a format error in a CALL statement.

CM Common Error

Indicates a format error in a COMMON statement.

CN Continuation Error

Indicates that more than nineteen continuation cards appear in succession or that one such card appears in an illogical sequence.

CO Common-Overflow Error

Indicates that the amount of COMMON storage, required by the main program or specified to the compiler, is less than that required by the current program or subroutine.

CT Continue Error

Indicates that a CONTINUE statement is missing a statement number.

DA Duplicate-Argument Error

Indicates that duplicate dummy arguments appear in a function-definition statement.

DC Decimal-Constant Error

Indicates a format error in the expression of a FORTRAN decimal constant.

DD Duplicate-Dimension Error

Indicates that a variable currently being dimensioned has been previously dimensioned.

DF Duplicate-Function-Name Error

Indicates that the function name in the current function-definition statement has occurred as the name of a previously defined function.

DM Dimension Error

Indicates a format error in a DIMENSION statement.

DO Do Error

Indicates a format error in a DO statement.

DP Duplicate-Statement-Number Error

Indicates that the current statement number has previously appeared in the statement-number field.

DS Missing-Do-Number Error

Indicates that references have been made in DO statements to statement numbers which did not appear somewhere in the statement-number field of a card.

DT Data Error

Indicates a format error in a DATA statement.

EC Equivalence-Contradiction Error

Indicates that a variable currently appearing in an EQUIVALENCE statement cannot be equivalenced because of an inherent contradiction in the statement.

EF End-of-File Error

Indicates than end-of-file card is detected before the last END card is encountered.

EM Exponential-Mode Error

Indicates that the mode of the base or the exponent of an indicated exponentiation process is improper.

EQ Equivalence Error

Indicates a format error in an EQUIVALENCE statement.

EX Exponent Error

Indicates a format error in the exponent portion of an indicated exponentiation process.

FL Function-List Error

Indicates a format error in an EXTERNAL statement or F-type statement.

FM Format Error

Indicates a format error in a statement whose type cannot be determined.

FN Format-Statement-Number Error

Indicates that a FORMAT statement is missing a statement number.

FS Format-Specification Error

Indicates a format error in the specification portion of a FORMAT statement.

FT Function-Type Error

Indicates a format error in a TYPE statement.

GO Go-To Error

Indicates a format error in a GO statement.

IF If Error

Indicates a format error in an IF statement.

IL Indexed-List Error

Indicates a format error in an indexed list of the current input/output statement.

LR Library-Reference Error

Indicates that a reference has been made to a standard library subroutine where more arguments appeared in the reference than are provided for by the subroutine.

LN Name-List Error

Indicates a format error or a reference to an array with variable dimensions in a NAMELIST statement.

LS List Error

Indicates a format error in an input/output list.

MA Misuse-of-Argument Error

Indicates that an argument of the SUBROUTINE or FUNCTION being compiled has been misused in an EQUIVALENCE statement.

MC Machine-Constant Error

Indicates an error in the formatting of a constant in a tag-defining line of coding involving the pseudo operations RES, COM, CON, ABS, SUB, or HOL.

MD Machine-Duplicate-Tag Error

Indicates that a tag appearing in a tag-defining line of coding involving the pseudo operations RES, COM, CON, ABS, SUB, or HOL has been previously defined.

MF Machine-Format Error

Indicates a format error in a tag-defining line of coding involving the pseudo operations RES, COM, CON, ABS, SUB, or HOL.

ML Machine-Location-Tag Error

Indicates that a tag appearing in an address field of a machine instruction did not appear in a subsequent location field.

MO Memory-Overflow Error

Indicates that the compiler field length, as specified on the job card, is too short.

MR Missing-Subroutine Error

Indicates that subroutines which are not in the standard subroutine library have been referenced by the Fortran program or its Fortran-coded subroutines.

MS Missing-Statement-Number Error

Indicates that references have been made to statement numbers which did not appear somewhere in the statement-number field of a card.

MT Machine-Tag-Definition Error

Indicates a format error in a tag-defining line of coding involving the pseudo operations RES, COM, CON, ABS, SUB, or HOL.

NC Name-Conflict Error

Indicates that the name of a SUBROUTINE or FUNCTION conflicts with prior usage of the name.

NM Name Error

Indicates an error in formatting of the name (header) card.

OD Dimension-Statement-Order Error

Indicates that a reference to an array has been made prior to the appearance of the array name in a DIMENSION statement.

PN Parentheses Error

Indicates that an unequal number of opened and closed parentheses appear in a statement.

RN Return Error

Indicates a format error in a RETURN statement.

SB Subscript Error

Indicates a format error in a subscript of an array reference currently being processed.

SE Sense Error

Indicates a format error in a SENSE statement.

SF Short-Field-Length Error

Indicates that the required field length of the program or subroutine being compiled exceeds the specified one.

SL Subroutine-Storage-Limit Error

Indicates that the compiler field length, as specified on the job card, is exceeded at the time that standard library subroutines are being assembled.

SM Statement-Number-Field Error

Indicates a format error in the statement-number field of the current statement.

SN Statement-Number Error

Indicates a format error in a position where a statement number should appear.

SY System Error

Indicates a machine malfunction.

TM Too-Many-Arguments Error

Indicates that a subroutine reference has more than sixty arguments or that the program or subroutine being compiled has more than sixty arguments.

TY Type Error

Indicates a format error in a TYPE statement.

UA Unidentified-Array Error

Indicates that a reference has been made to an array where the array name has not previously appeared in a DIMENSION statement.

UE Unidentified-Equipment Error

Indicates a reference to an input/output file which was not listed in the header card of the main program.

US Unreferenced-Binary-Subroutine Error

Indicates that a subroutine on binary cards following the Fortran program and its Fortran-coded subroutines has not been referenced.

VC Variable-Name-Conflict Error

Indicates that a variable name appears which conflicts with some other prior usage.

VD Variable-Dimensioned-Array Error

Indicates that an array whose dimensions are arguments to the SUBROUTINE or FUNCTION being compiled has been misused.

VN Variable-Name Error

Indicates an error in a variable name appearing in the current statement.

XF Expression-Format Error

Indicates a format error in the expression currently being processed.

XM Expression-Mode Error

Indicates that an expression currently being processed is one in which modes are mixed.

Error Messages Entered into Dayfile

The following error statements may be entered into the dayfile during execution of a FORTRAN program, or a machine language program using the FORTRAN subroutines.

After each message the program is aborted.

Origin

INPUTC - data input subroutine.

INP(FN) FN is the number of the format statement being executed at time of error.

END OF FILE, DATA INPUT . Attempt made to read past file mark.

OUTPTC - data output subroutine.

BCD OUTPUT ** UNASSIGNED MEDIUM . Failure to specify a file name during compilation.

NO DEC. PT IN E OR F . No significance specified in E or F conversion.

EXCEEDED RECORD LENGTH . More than 150 characters have been entered as a record.

ILLEGAL FUNCTIONAL LETTER . Illegal format letter.

PAREN. LEVEL N(N(. Format has parenthetical group within a parenthetical group.

For OUTPTC messages, the format number is entered in the line following the message.

INPUTB - binary input.

BIN INPUT ** UNASSIGNED MEDIUM.

OUTPTB - binary output.

BIN OUTPUT ** UNASSIGNED MEDIUM.

ENDFIL - write end of file.

EOF ** UNASSIGNED MEDIUM.

The Central Processor Assembler provides a language for writing programs and subroutines that may be executed independently or as a FORTRAN subroutine. The basic features of the language were originally described in Volume I ASCENT of CONTROL DATA® 6600 Programming System/Reference Manual.

Assemblies are initiated by calling for CPUASM from the systems library. Punch, list, and execute options are parameters to CPUASM. The binary version is always transferred to the disk file regardless of option parameters or assembly errors. P, L, and X are the three parameters to the assembler:

P - punch binary cards.
L - list side by side.
X - execute program.

Example of program to be assembled:

```
JOB,5,3,30000.
CPUASM(L,X,P)
7,8,9
PROGRAM  NAME
  STATEMENTS
END
6,7,8,9
```

If the execute option (X) is deleted, the program may still be called by referencing the program via the control card following the assembly.

Symbols:

Symbols may be any arrangement of characters up to 10 characters in length. A symbol must begin with an alphabetic letter and must not contain any separators. The special characters used as separators are + , - / * = $ and SPACE. Symbols used in address fields may be signed + or - allowing them to be added to one another or to a constant.

Constants:

Constants may be either octal or decimal fixed point integers $\leq 2^{60}-1$, a symbolic address or a single precision floating point number. Integers are assumed to be decimal unless appended by the letter B. Floating point numbers must have a decimal point and may optionally be followed by an exponent E±N.

Field definitions:

The location field starts in column 2 and may be 1-10 characters in length. If a location tag is present the operation code must be separated from it by a space code or must not start prior to column 7 if no location tag is present.

The address field is variable in length and is separated from the operation code. In general, the address field may be any combination of terms, integer and/or symbolic, as long as the constant field for the instruction is not exceeded. Terms in address fields may also be a combination of 1 or 2 A, B, or X registers; if only 1 term is present Bo is assumed as the second term.

Pseudo Operation Codes.

PROGRAM:	Defines the job to be a program; the symbol in the address field is the name of the program as referenced by the system.
SUBROUTINE:	Defines the job to be a subroutine and sets relocatable bits for later use by the FORTRAN compiler or machine program. The symbol in the address field is the name used to reference the subroutine.
END:	Last card of a program or subroutine.
BSS:	Address field defines the length of the block reservation. Address field may be integer constant or symbolic constant.
EQU:	Equivalences a symbol to another symbol or a constant.
DPC:	Allows the entry of console display codes, the length being specified by a 2 digit integer, or the codes between *'s are entered.
BCD:	Converts the characters enclosed by the asterisks or converts the characters specified by the beginning 2 digit integer.
CON:	Converts each term to a 60 bit constant.
EJECT:	Ejects the listing to the top of the next page.
SPACE:	Spaces the number of lines specified by the address field.

OPERATION CODES

OP	MNEMONIC	ADDRESS	REMARKS
			.Branch unit
00	PS		.Program stop
01	RJ	K	.Return jump to K
02	JP	Bi+ K	.Jump to Bi+K
030	ZR	Xi K	.Jump to K if Xi=0
031	NZ	Xi K	.Jump to K if Xi≠0
032	PL	Xi K	.Jump to K if Xi=plus (positive)
033	NG	Xi K	.Jump to K if Xi=negative
034	IR	Xi K	.Jump to K if Xi is in range
035	OR	Xi K	.Jump to K if Xi is out of range
036	DF	Xi K	.Jump to K if Xi is definite
037	ID	Xi K	.Jump to K if Xi is indefinite
04	EQ	BiBjK	.Jump to K if Bi=Bj
04	ZR	Bi K	.Jump to K if Bi=B0
05	NE	BiBjK	.Jump to K if Bi≠Bj
05	NZ	Bi K	.Jump to K if Bi≠B0
06	GE	BiBjK	.Jump to K if Bi≥Bj
06	PL	Bi K	.Jump to K if Bi≥B0
07	LT	BiBjK	.Jump to K if Bi<Bj
07	NG	Bi K	.Jump to K if Bi<B0
			.Boolean Unit
10	BXi	Xj	.Transmit Xj to Xi
11	BXi	Xj*Xk	.Logical Product of Xj & Xk to Xi
12	BXi	Xj+Xk	.Logical sum of Xj & Xk to Xi
13	BXi	Xj-Xk	.Logical difference of Xj & Xk to Xi
14	BXi	-Xk	.Transmit the comp. of Xj to Xi
15	BXi	-Xk*Xj	.Logical product of Xj & Xk comp. to Xi
16	BXi	-Xk+Xj	.Logical sum of Xj & Xk comp. to Xi
17	BXi	-Xk-Xj	.Logical difference of Xj & Xk comp. to Xi
			.Shift Unit
20	LXi	jk	.Left shift Xi,jk places
21	AXi	jk	.Arithmetic right shift Xi,jk places
22	LXi	Bj,Xk	.Left shift Xi nominally Bi places
23	AXi	Bj,Xk	.Arithmetic right shift Xi nominally Bi places
24	NXi	Bj Xk	.Normalize Xk in Xi and Bj
25	ZXi	Bj Xk	.Round and normalize Xk in Xi and Bj
26	UXi	Bj Xk	.Unpack Xk to Xi and Bj
27	PXi	Bj Xk	.Pack Xi from Xk and Bj
43	MXi	jk	.Form mask in Xi,jk bits
			.Add Unit
30	FXi	Xj+Xk	.Floating sum of Xj and Xk to Xi
31	FXi	Xj-Xk	.Floating difference Xj and Xk to Xi
32	DXi	Xj+Xk	.Floating DP sum of Xj and Xk to Xi
33	DXi	Xj-Xk	.Floating DP difference of Xj and Xk to Xi
34	RXi	Xj+Xk	.Round floating sum of Xj and Xk to Xi

OP	MNEMONIC	ADDRESS	REMARKS
35	RXi	Xj-Xk	.Round floating difference of Xj and Xk to Xi .Long Add Unit
36	IXi	Xj+Xk	.Integer sum of Xj and Xk to Xi
37	IXi	Xj-Xk	.Integer difference of Xj and Xk to Xi .Multiply Unit
40	FXi	Xj*Xk	.Floating product of Xj and Xk to Xi
41	RXi	Xj*Xk	.Round floating product of Xj & Xk to Xi
42	DXi	Xj*Xk	.Floating DP product of Xj & Xk to Xi .Divide Unit
44	FXi	Xj/Xk	.Floating divide Xj by Xk to Xi
45	RXi	Xj/Xk	.Round floating divide Xj by Xk to Xi
46	NO		.No operation
47	CXi	Xj	.Count the number of 1's in Xj to Xi .Increment Unit
50	SAi	Aj+K	.Set Ai to Aj+K
50	SAi	Aj-K	.Set Ai to Aj+ comp. of K
51	SAi	Bj+K	.Set Ai to Bj+K
51	SAi	Bj-K	.Set Ai to Bj+ comp. of K
52	SAi	Xj+K	.Set Ai to Xj+K
52	SAi	Xj-K	.Set Ai to Xj+ comp. of K
53	SAi	Xj+Bk	.Set Ai to Xj+Bk
54	SAi	Aj+Bk	.Set Ai to Aj+Bk
55	SAi	Aj-Bk	.Set Ai to Aj-Bk
56	SAi	Bj+Bk	.Set Ai to Bj+Bk
57	SAi	Bj-Bk	.Set Ai to Bj-Bk
60	SBi	Aj+K	.Set Bi to Aj+K
60	SBi	Aj-K	.Set Bi to Aj+ comp. of K
61	SBi	Bj+K	.Set Bi to Bj+K
61	SBi	Bj-K	.Set Bi to Bj+ comp. of K
62	SBi	Xj+K	.Set Bi to Xj+K
62	SBi	Xj-K	.Set Bi to Xj+ comp. of K
63	SBi	Xj+Bk	.Set Bi to Xj+Bk
64	SBi	Aj+Bk	.Set Bi to Aj+Bk
65	SBi	Aj-Bk	.Set Bi to Aj-Bk
66	SBi	Bj+Bk	.Set Bi to Bj+Bk
67	SBi	Bj-Bk	.Set Bi to Bj-Bk
70	SXi	Aj+K	.Set Xi to Aj+K
70	SXi	Aj-K	.Set Xi to Aj+ comp. of K
71	SXi	Bj+K	.Set Xi to Bj+K
71	SXi	Bj-K	.Set Xi to Bj+ comp. of K
72	SXi	Xj+K	.Set Xi to Xj+K
72	SXi	Xj-K	.Set Xi to Xj+ comp. of K
73	SXi	Xj+Bk	.Set Xi to Xj+Bk
74	SXi	Aj+Bk	.Set Xi to Aj+Bk
75	SXi	Aj-Bk	.Set Xi to Aj-Bk
76	SXi	Bj+Bk	.Set Xi to Bj+Bk
77	SXi	Bj-Bk	.Set Xi to Bj-Bk

Error Codes.

The following error codes may appear on the side by side listing:

P - Location symbol has previously been defined.
U - Symbol in address field is undefined.
C - Constant out of range or contains illegal codes.
I - Instruction field not legal.
M - Symbol in address field is multiply defined.
O - Address field does not correspond to any valid instruction.

This assembler is a PPU program which converts PPU symbolic language into PPU absolute language.

The assembler produces a side by side listing and binary cards. Punching of cards is suppressed if an error has been detected in the assembled program. The binary cards punched may be either 80-column binary cards or Chippewa Laboratory standard binary card format, i.e.: Columns 1 and 2 contain a binary card identifier (7-9 punch), number of 60-bit words on the card (15_{10} maximum) and a card check sum. Column 80 is a card sequence number. To punch Chippewa Laboratory binary cards, the job cards are:

```
Job (1, 100, 1000)
PAS
789
Program Deck
6789
```

To punch standard 80-column binary cards, the job cards are:

```
Job (1, 100, 1000)
PAS, 1000
789
Program Deck
6789
```

There is no provision for leaving a running version of the assembled program in the computer. The assembled program is available as a card deck and/or a line printer listing.

CARD FORMAT:

Card format for this assembler is fixed field except for the remarks card which may be one of two variations.

(1) Pseudo op "REM" which must appear in the OP field (Cols. 6-7-8) is free fielded thereafter and

(2) * in Column 1, and free fielded thereafter.

The other exception is the "DIS" pseudo op which converts all characters following DIS to display code.

The standard format then, used in all other instances, is as follows:

Col. 1-4 = Location field. Maximum of 4 alpha-numeric characters.

Col. 5 = Blank.

Col. 6-8 = Op field. Three character mnemonic (See Table 1).

Col. 9 = Blank.

Col. 10-13 = Tag field. Maximum of 4 alpha-numeric characters. There are no provisions for adding a constant to or subtracting a constant from a tag.

Col. 14-19 = Blank.

Col. 20-80 = Comments.

CONSTANTS:

Numeric constants used in a program will be placed in Columns 10-13 and are considered to be octal. They therefore have a range of 0000 to 7777 in a two address instruction or a range of 00 to 77 in a one address instruction. All constants are right adjusted by the assembler.

Constants may also be in the form of a tag; however, the converted value must also follow the rules for one and two address instructions.

OP CODES AND PSEUDO OPS:

These are defined in Table 1.

ERROR FLAGS:

There are several errors which will be detected by the assembler and shown as a 1 character/error tag in the left margin of the assembled listing.

Ø = Illegal Op code.

U = Undefined tag has been referenced.

M = Location tag used more than once.

R = Range error. d portion is greater than 77 or on a jump instruction (03-07) location tag is found to be more than 37_8 forward or backward.

TABLE 1. MNEMONIC OP CODES AND PSEUDO OPS

Op Codes

Mnemonic & Octal Code	Name	Mnemonic & Octal Code	Name
PSN 00	Pass	LMI 43	Logical difference ((d))
LJM 01	Long jump to m + (d)	STI 44	Store ((d))
RJM 02	Return jump to m + (d)	RAI 45	Replace add ((d))
UJN 03	Unconditional jump d	AOI 46	Replace add one ((d))
ZJN 04	Zero jump d	SOI 47	Replace subtract one ((d))
NJN 05	Nonzero jump d		
PJN 06	Plus jump d	LDM 50	Load (m + (d))
MJN 07	Minus jump d	ADM 51	Add (m + (d))
SHN 10	Shift d	SBM 52	Subtract (m + (d))
LMN 11	Logical difference d	LMM 53	Logical Difference (m + (d))
LPN 12	Logical product d	STM 54	Store (m + (d))
SCN 13	Selective clear d	RAM 55	Replace add (m + (d))
LDN 14	Load d	AOM 56	Replace add one (m + (d))
LCN 15	Load complement d	SOM 57	Replace subtract one (m + (d)
ADN 16	Add d		
SBN 17	Subtract d	CRD 60	Central read from (A) to d
		CRM 61	Central read (d) words from (A) to m
LDC 20	Load dm	CWD 62	Central write to (A) from d
ADC 21	Add dm	CWM 63	Central write (d) words to (A) from m
LPC 22	Logical product dm		
LMC 23	Logical difference dm		
		AJM 64	Jump to m if channel d active
PSN 24	Pass		
PSN 25	Pass	IJM 65	Jump to m if channel d inactive
		FJM 66	Jump to m if channel d full
EXN 26	Exchange jump		
RPN 27	Read program address	EJM 67	Jump to m if channel d empty
LDD 30	Load (d)	IAN 70	Input to A from channel d
ADD 31	Add (d)	IAM 71	Input (A) words to m from channel d
SBD 32	Subtract (d)		
LMD 33	Logical difference (d)	OAN 72	Output from A on channel d
STD 34	Store (d)	OAM 73	Output (A) words from m on channel d
RAD 35	Replace add (d)		
AOD 36	Replace add one (d)	ACN 74	Activate channel d
SOD 37	Replace subtract one (d)	DCN 75	Disconnect channel d
		FAN 76	Function (A) on channel d
LDI 40	Load ((d))	FNC 77	Function m on channel d
ADI 41	Add ((d))		
SBI 42	Subtract ((d))		

TABLE 1. (cont.)

Pseudo Ops

IDT = Name of program.

ØRG = Beginning location of assembled program. Will be set to 1000 if no ØRG specified.

SBL = Set beginning address of binary punch out.

BLR = Block locations reserve.

REM = Remarks card. * in Column 1 is also a remarks card.

EQU = Equates a tag to a numeric constant or a pre-defined location tag.

DIS = Converts the remaining characters to display code and packs them 2 to a word. Word following packed display code is all zeros.

END = Defines the last card of the deck to be assembled.

PERIPH
PERIPHERAL ASSEMBLY LANGUAGE

PERIPH is identical to PAS except that it is run by the CPU rather than a PPU.

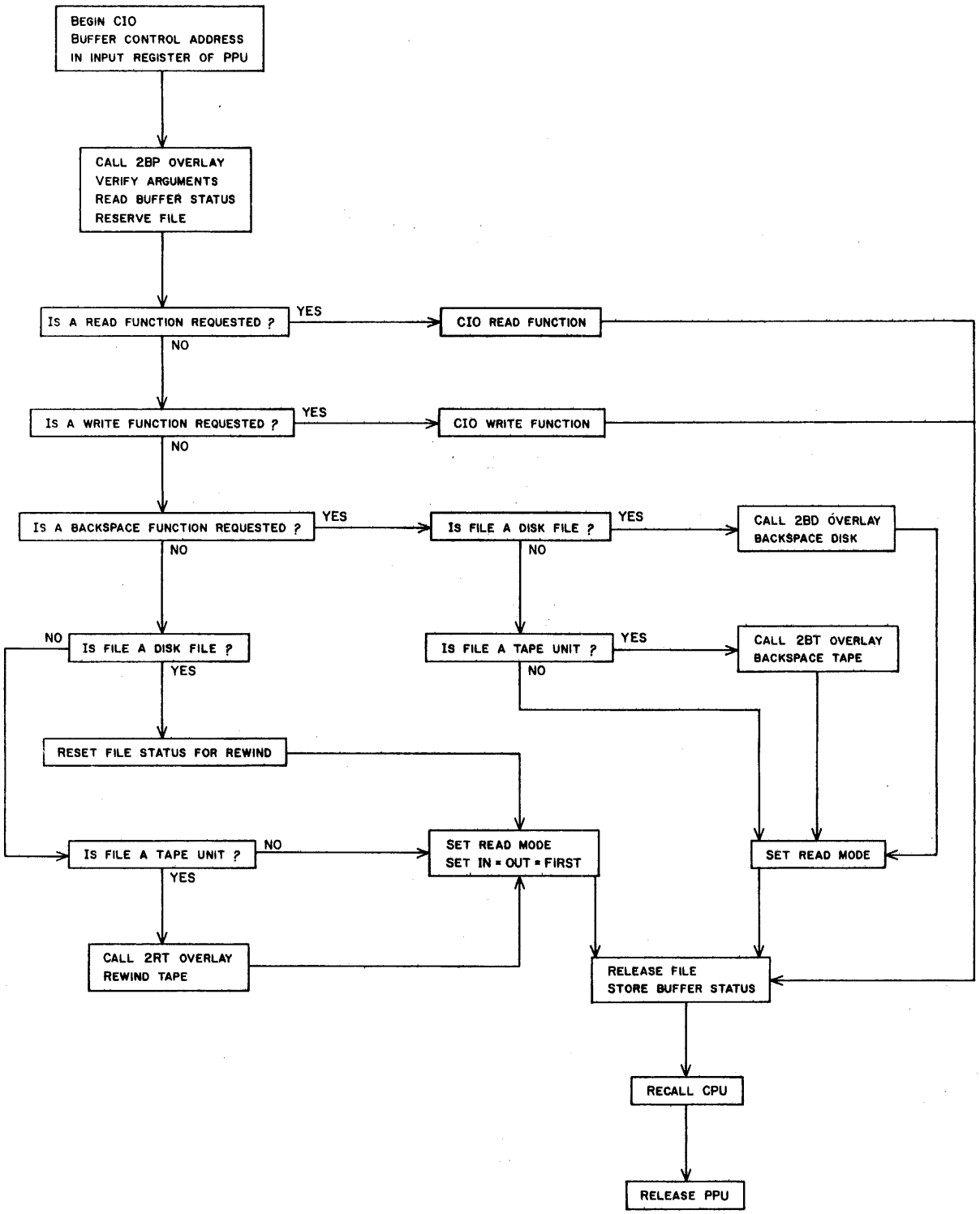
Card Image Records - Second File Library Tape

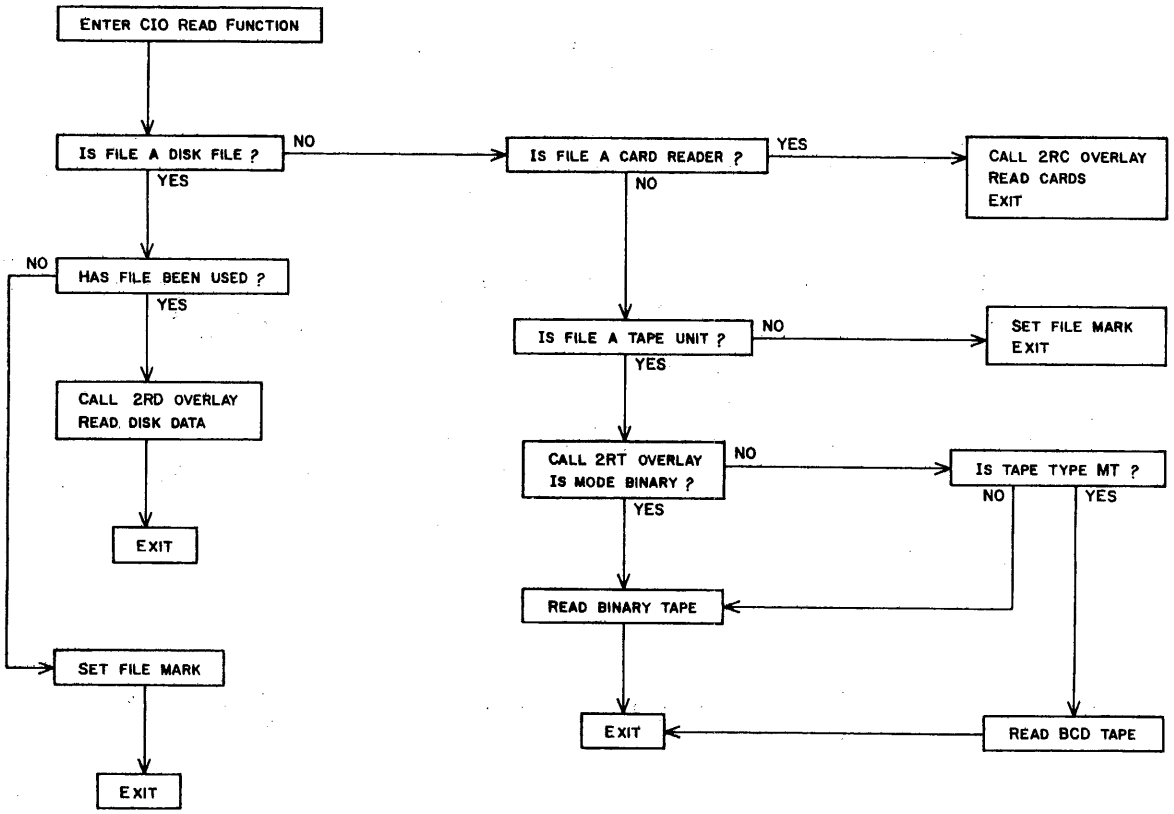
The following routines are listed in card image form in the second file on the library tape. Each routine is entered as one record.

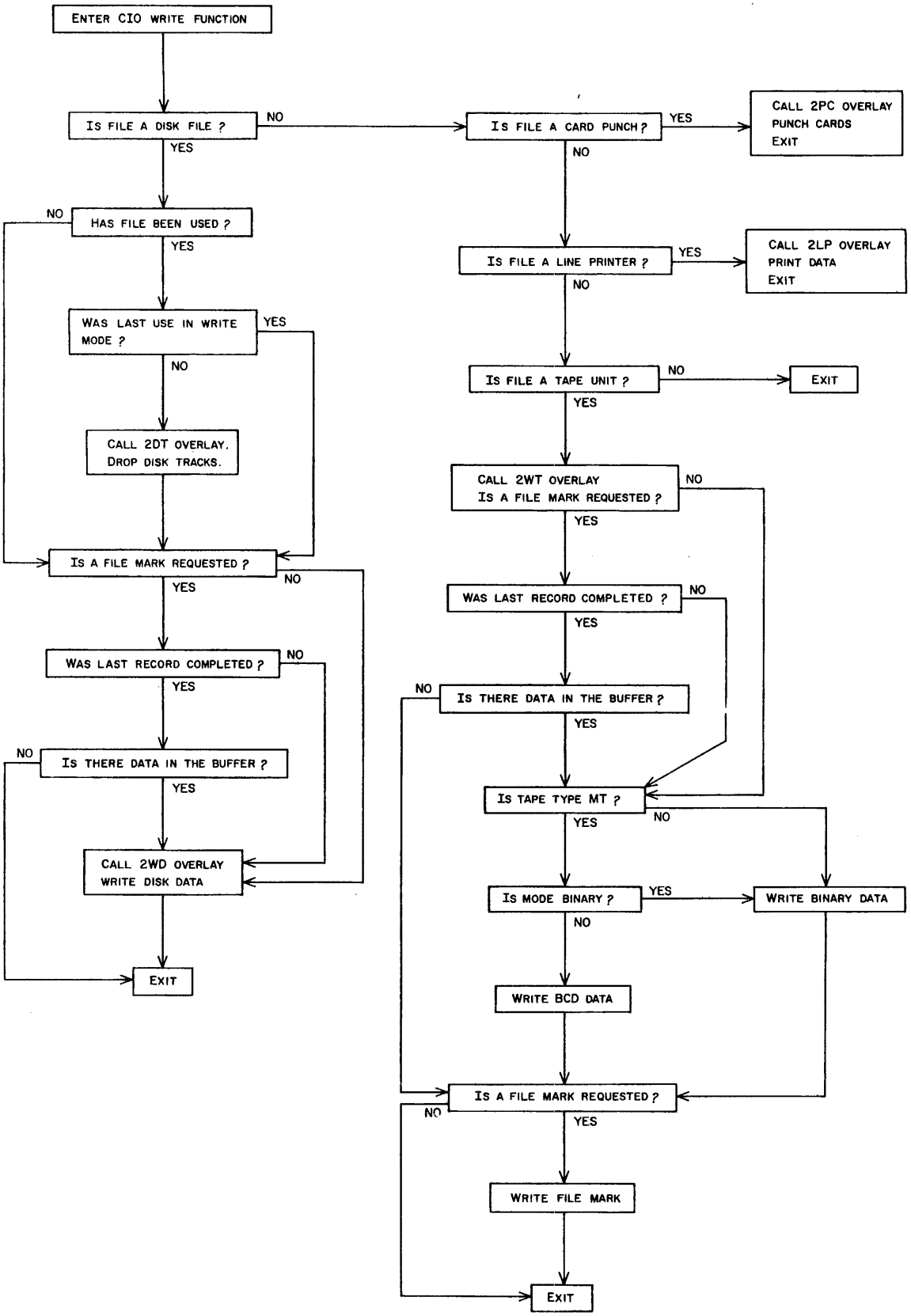
(00) Loader
(01) DSD
(02) MTR
(03) CR
(04) 1AJ
(05) 1BJ
(06) 1DJ
(07) 1LJ
(08) 1LT
(09) 1TD
(10) 2BD
(11) 2BP
(12) 2BT
(13) 2DF
(14) 2DT
(15) 2EF
(16) 2LP
(17) 2PC
(18) 2RC
(19) 2RD
(20) 2RT
(21) 2SD
(22) 2TJ
(23) 2TS
(24) 2WD
(25) 2WT
(26) CIO
(27) CLL
(28) DIS
(29) DMP
(30) EXU
(31) LBC
(32) LOC
(33) MSG
(34) PBC
(35) BKSP
(36) COPY
(37) COPYBF
(38) COPYBR
(39) COPYCF
(40) COPYCR
(41) REWIND
(42) VERIFY
(43) APRAB
(44) ALOG
(45) ALOG10
(46) ATAN
(47) COS
(48) DVCHK
(49) END
(50) EXIT
(51) EXP
(52) IBAIEX
(53) IFENDF
(54) OVERFL
(55) PAUSE
(56) RBAIEX
(57) RBAREX
(58) REMARK
(59) SIN
(60) SQRT
(61) SLITE
(62) SLITET
(63) SSWTCH
(64) START
(65) STOP
(66) TAN
(67) TANH
(68) TIME
(69) COPYSBF
(70) RANF
(71) -
(72) -
(73) -
(74) -
(75) -
(76) -
(77) XLOCF
(78) INPUTB
(79) INPUTC
(80) OUTPTB
(81) OUTPTC
(82) ENDFIL
(83) REWINM
(84) BACKSP
(85) DISPLA
(86) 007
(87) RUN

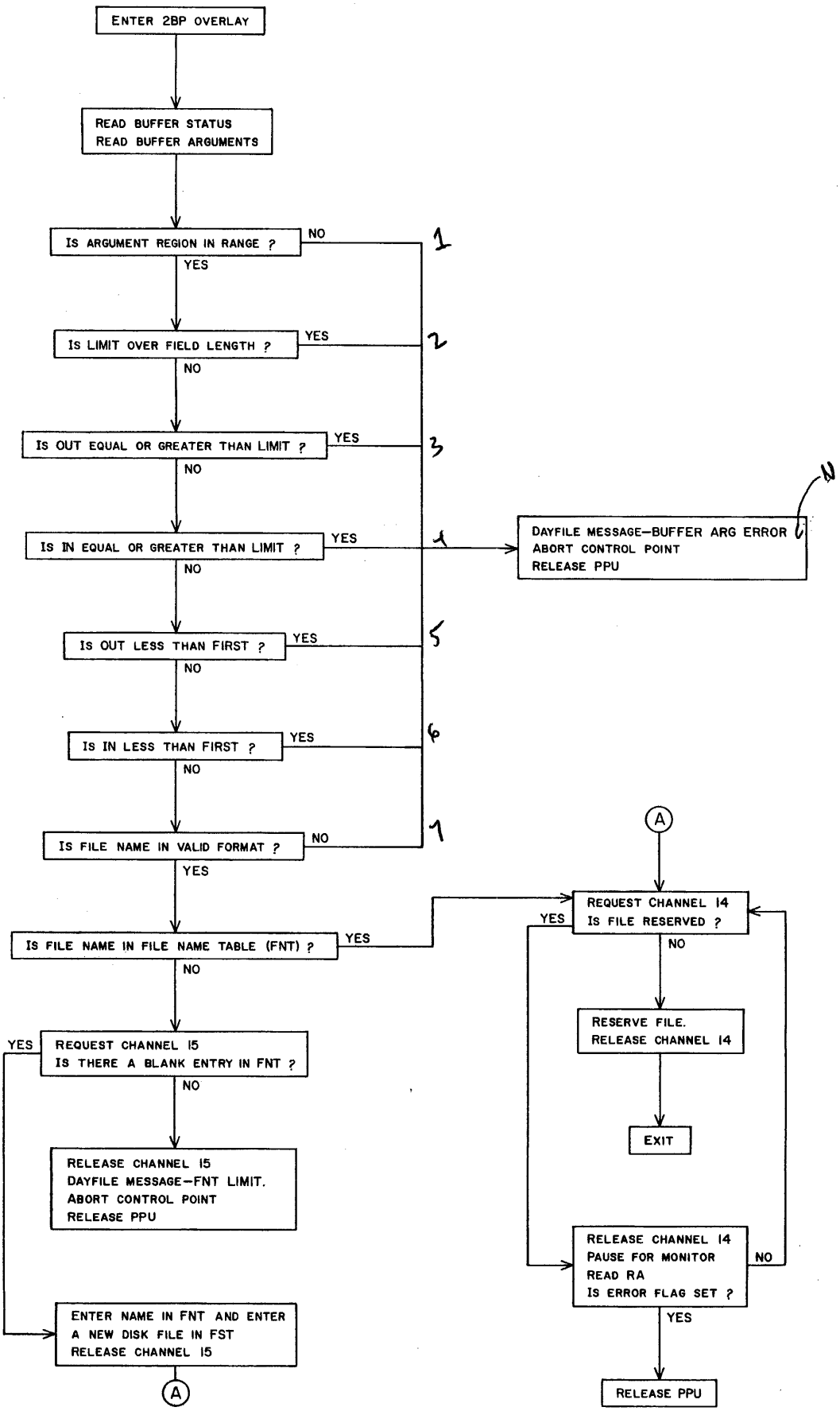
APPENDIX A

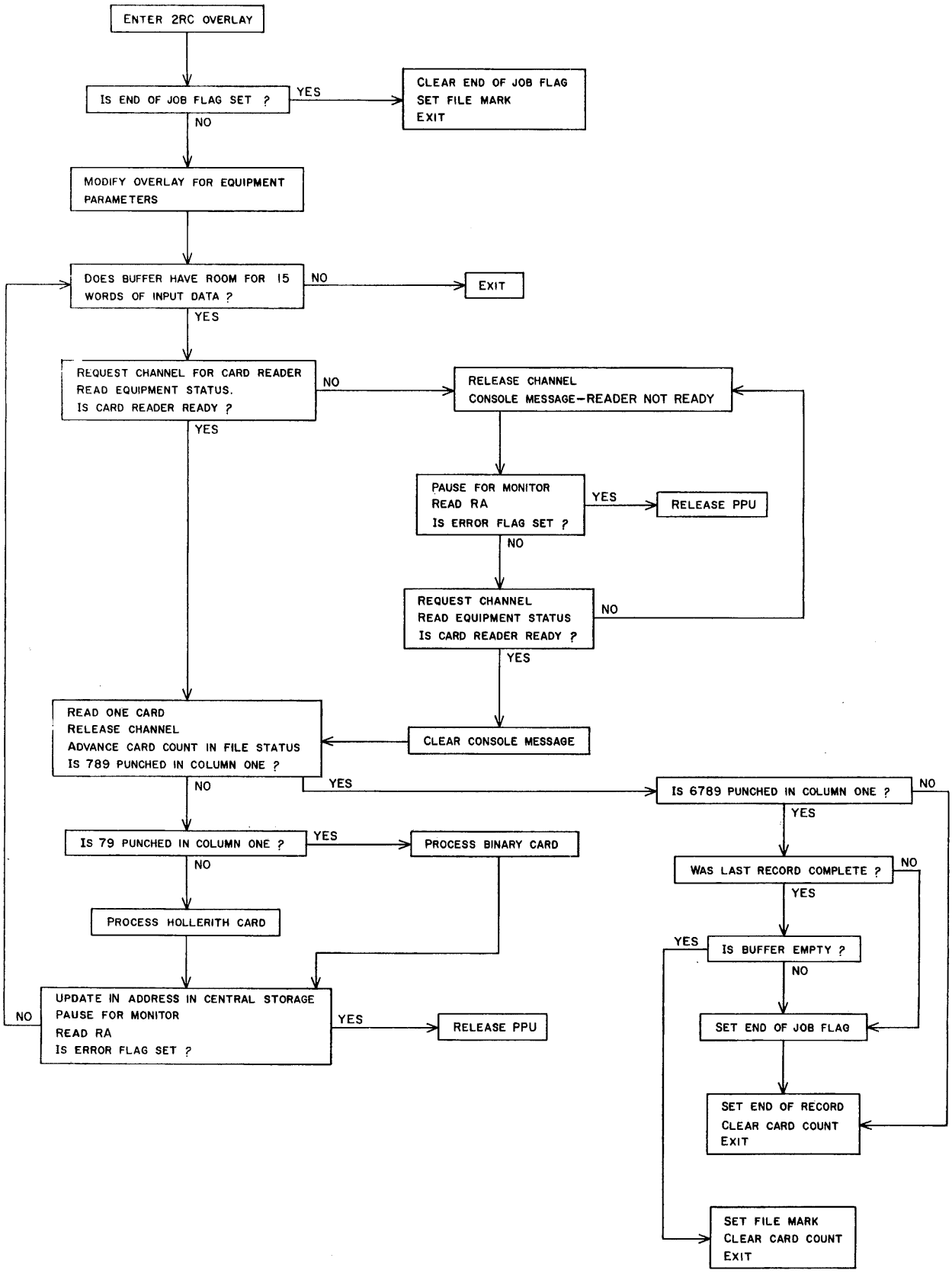
CHIPPEWA OPERATING SYSTEM FLOW CHARTS

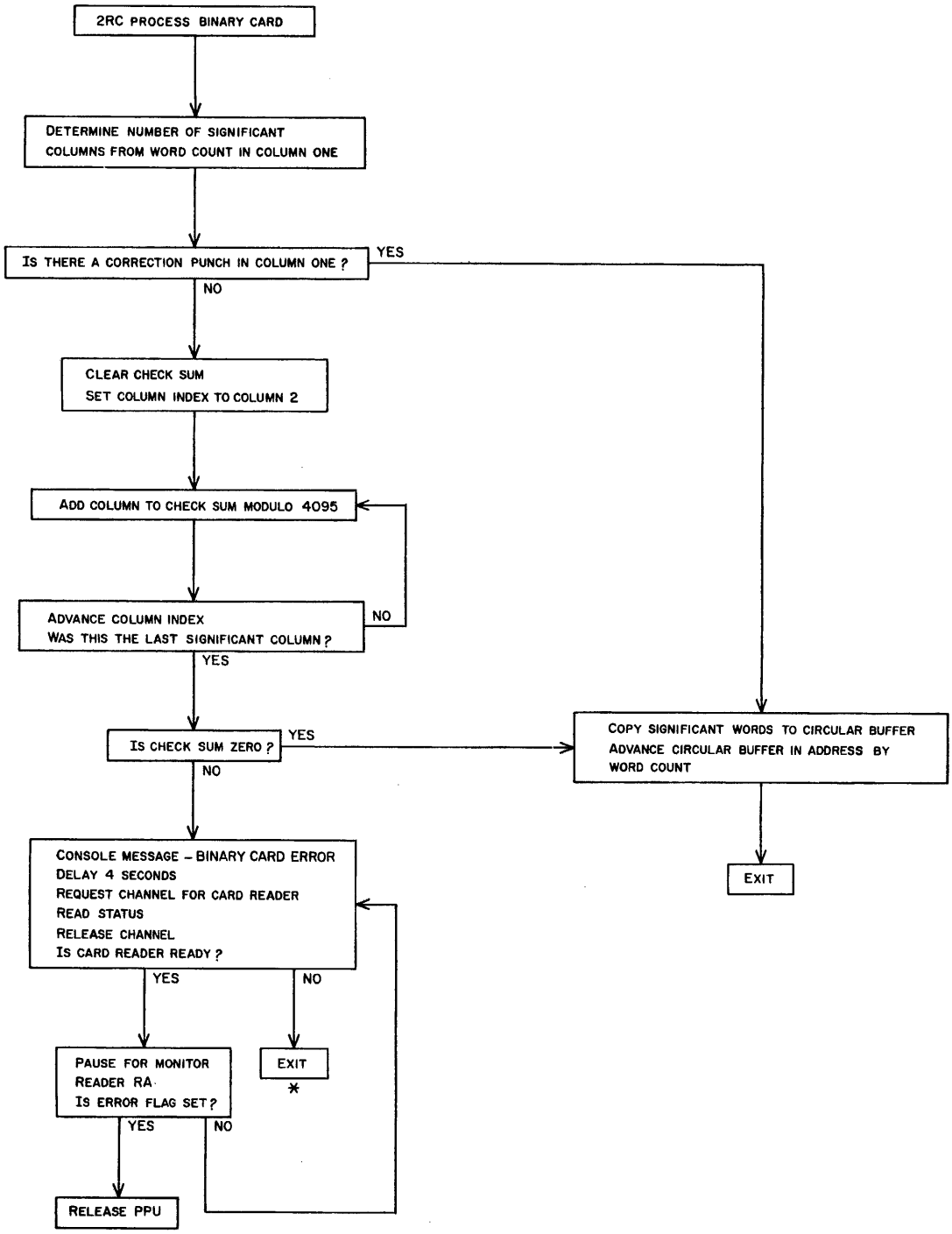

BEGIN CIO
BUFFER CONTROL ADDRESS
IN INPUT REGISTER OF PPU
CALL 2BP OVERLAY
VERIFY ARGUMENTS
READ BUFFER STATUS
RESERVE FILE
IS A READ FUNCTION REQUESTED ?
YES
CIO READ FUNCTION
NO
IS A WRITE FUNCTION REQUESTED ?
YES
CIO WRITE FUNCTION
NO
IS A BACKSPACE FUNCTION REQUESTED ?
YES
IS FILE A DISK FILE ?
YES
CALL 2BD OVERLAY
BACKSPACE DISK
NO
NO
IS FILE A DISK FILE ?
IS FILE A TAPE UNIT ?
YES
CALL 2BT OVERLAY
BACKSPACE TAPE
YES
NO
RESET FILE STATUS FOR REWIND
IS FILE A TAPE UNIT ?
NO
SET READ MODE
SET IN = OUT = FIRST
SET READ MODE
YES
CALL 2RT OVERLAY
REWIND TAPE
RELEASE FILE
STORE BUFFER STATUS
RECALL CPU
RELEASE PPU


ENTER CIO READ FUNCTION
IS FILE A DISK FILE ?
NO
YES
IS FILE A CARD READER ?
YES
CALL 2RC OVERLAY
READ CARDS
EXIT
NO
HAS FILE BEEN USED ?
NO
YES
IS FILE A TAPE UNIT ?
NO
SET FILE MARK
EXIT
YES
CALL 2RD OVERLAY
READ DISK DATA
CALL 2RT OVERLAY
IS MODE BINARY ?
NO
IS TAPE TYPE MT ?
NO
YES
YES
EXIT
READ BINARY TAPE
SET FILE MARK
EXIT
EXIT
READ BCD TAPE

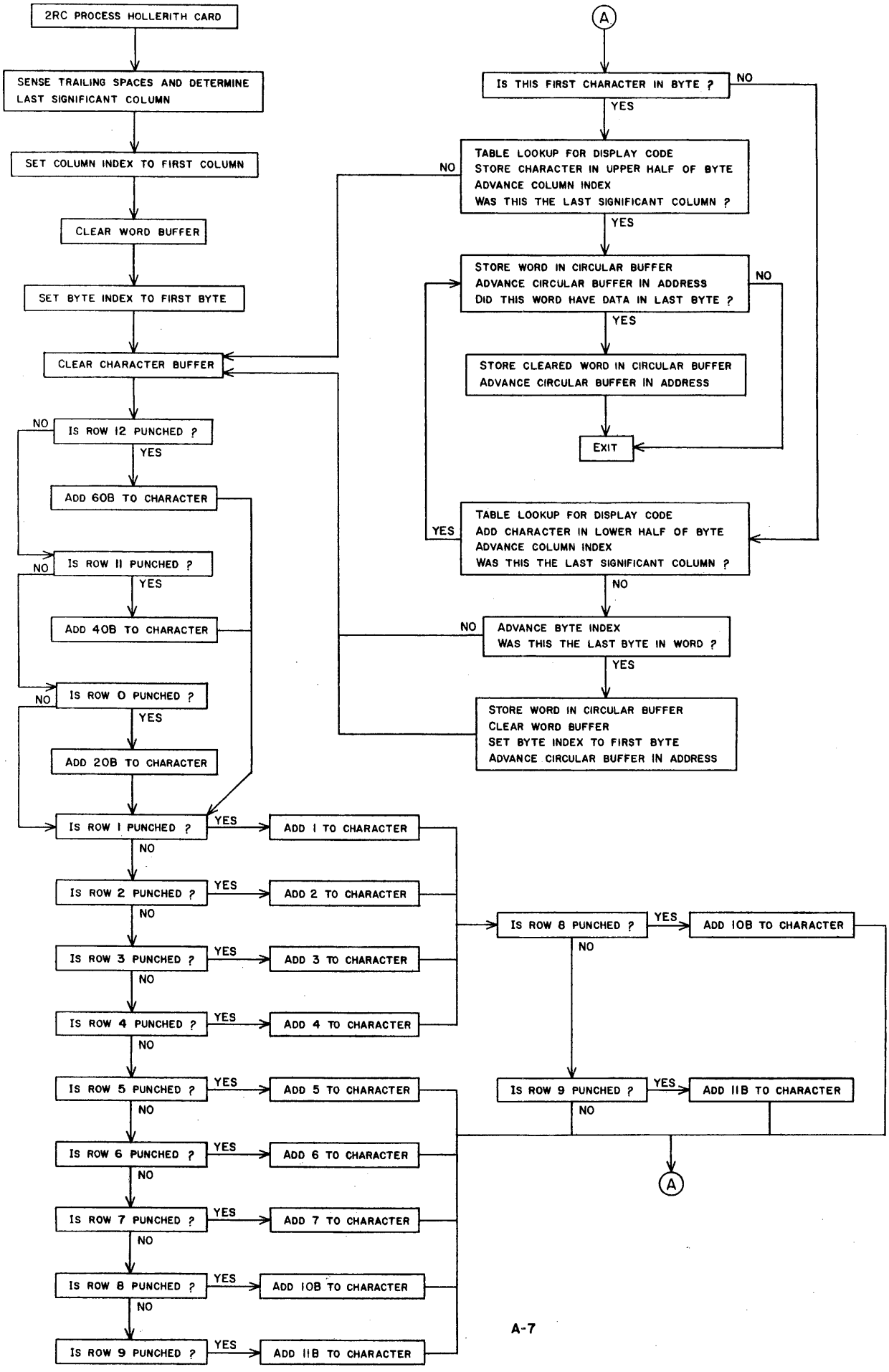

ENTER CIO WRITE FUNCTION
IS FILE A DISK FILE ?
NO
YES
IS FILE A CARD PUNCH ?
YES
CALL 2PC OVERLAY
PUNCH CARDS
EXIT
NO
HAS FILE BEEN USED ?
NO
YES
IS FILE A LINE PRINTER ?
YES
CALL 2LP OVERLAY
PRINT DATA
EXIT
NO
WAS LAST USE IN WRITE MODE ?
YES
NO
IS FILE A TAPE UNIT ?
NO
EXIT
YES
CALL 2DT OVERLAY.
DROP DISK TRACKS.
CALL 2WT OVERLAY
IS A FILE MARK REQUESTED ?
NO
YES
IS A FILE MARK REQUESTED ?
NO
YES
WAS LAST RECORD COMPLETED ?
NO
YES
WAS LAST RECORD COMPLETED ?
NO
YES
NO
IS THERE DATA IN THE BUFFER ?
YES
NO
IS THERE DATA IN THE BUFFER ?
YES
IS TAPE TYPE MT ?
YES
NO
CALL 2WD OVERLAY
WRITE DISK DATA
IS MODE BINARY ?
YES
WRITE BINARY DATA
NO
WRITE BCD DATA
IS A FILE MARK REQUESTED ?
NO
YES
WRITE FILE MARK
EXIT

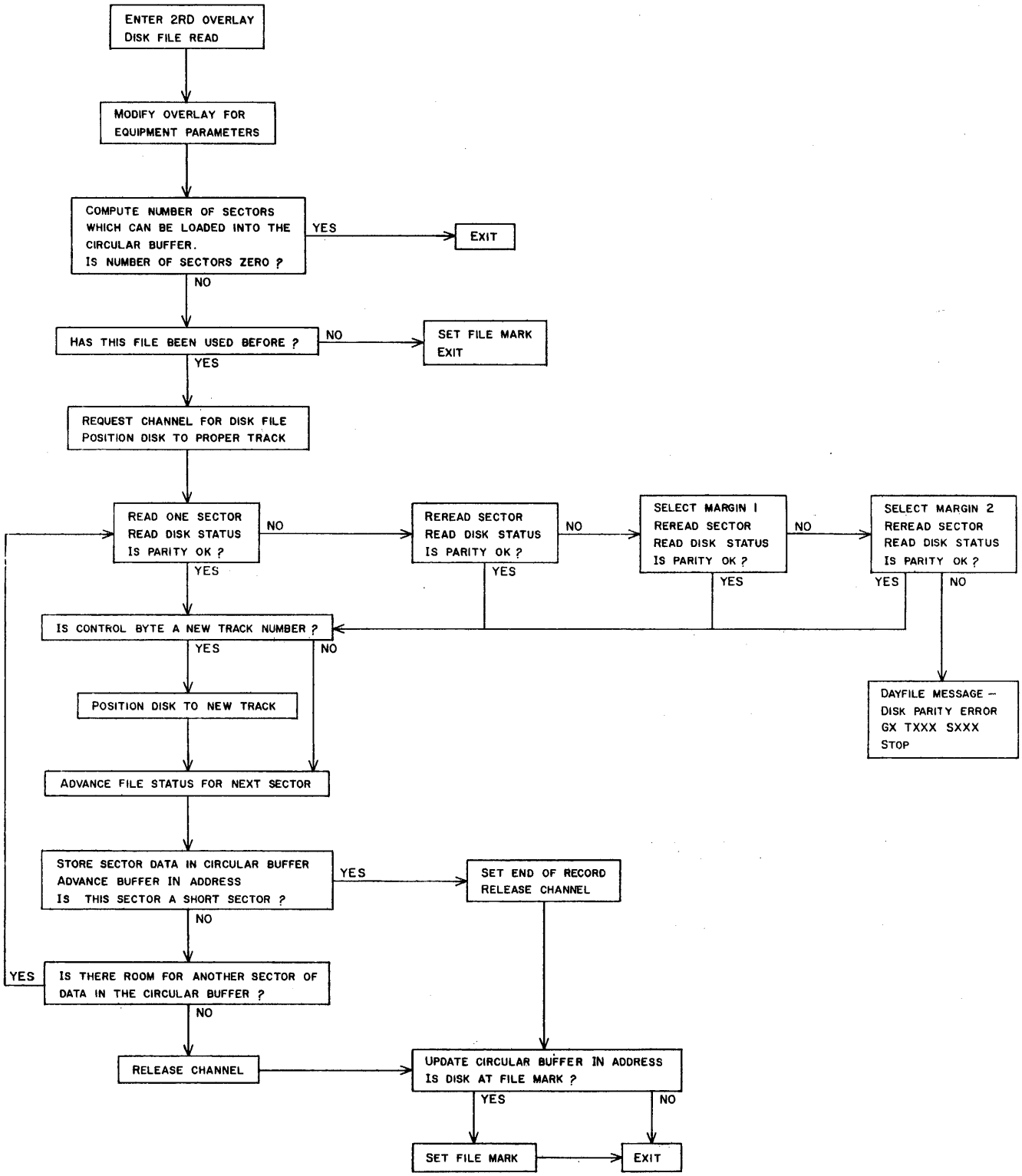

ENTER 2BP OVERLAY
READ BUFFER STATUS
READ BUFFER ARGUMENTS
IS ARGUMENT REGION IN RANGE ?
NO
1
YES
IS LIMIT OVER FIELD LENGTH ?
YES
2
NO
IS OUT EQUAL OR GREATER THAN LIMIT ?
YES
3
NO
IS IN EQUAL OR GREATER THAN LIMIT ?
YES
4
DAYFILE MESSAGE—BUFFER ARG ERROR
ABORT CONTROL POINT
RELEASE PPU
NO
IS OUT LESS THAN FIRST ?
YES
5
NO
IS IN LESS THAN FIRST ?
YES
6
NO
IS FILE NAME IN VALID FORMAT ?
NO
7
YES
IS FILE NAME IN FILE NAME TABLE (FNT) ?
YES
NO
YES
REQUEST CHANNEL 15
IS THERE A BLANK ENTRY IN FNT ?
NO
RELEASE CHANNEL 15
DAYFILE MESSAGE—FNT LIMIT.
ABORT CONTROL POINT
RELEASE PPU
ENTER NAME IN FNT AND ENTER
A NEW DISK FILE IN FST
RELEASE CHANNEL 15
A
A
REQUEST CHANNEL 14
IS FILE RESERVED ?
YES
NO
RESERVE FILE.
RELEASE CHANNEL 14
EXIT
RELEASE CHANNEL 14
PAUSE FOR MONITOR
READ RA
IS ERROR FLAG SET ?
NO
YES
RELEASE PPU

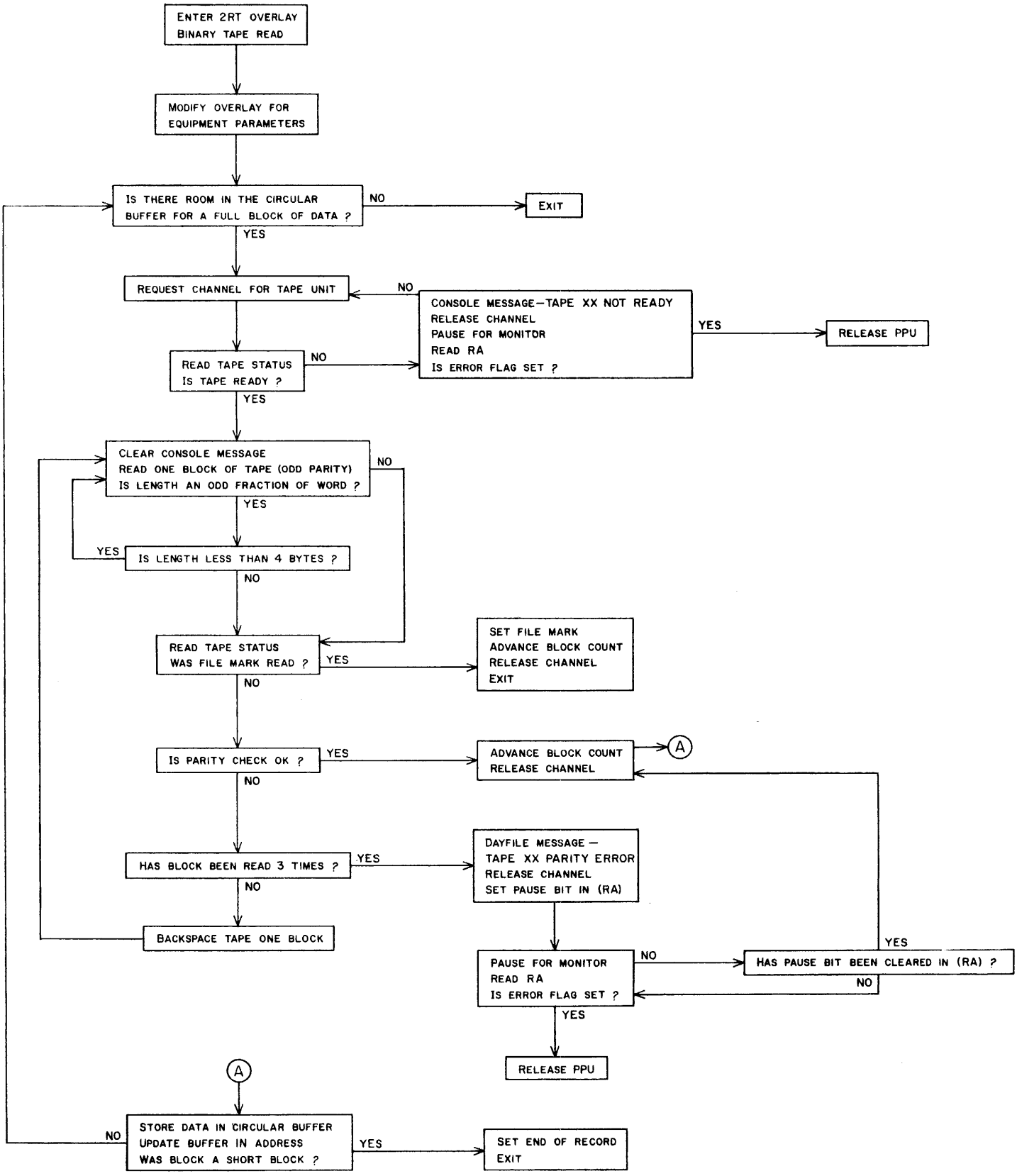

ENTER 2RC OVERLAY
IS END OF JOB FLAG SET ?
YES
CLEAR END OF JOB FLAG
SET FILE MARK
EXIT
NO
MODIFY OVERLAY FOR EQUIPMENT PARAMETERS
DOES BUFFER HAVE ROOM FOR 15 WORDS OF INPUT DATA ?
NO
EXIT
YES
REQUEST CHANNEL FOR CARD READER
READ EQUIPMENT STATUS.
IS CARD READER READY ?
NO
RELEASE CHANNEL
CONSOLE MESSAGE-READER NOT READY
YES
PAUSE FOR MONITOR
READ RA
IS ERROR FLAG SET ?
YES
RELEASE PPU
NO
REQUEST CHANNEL
READ EQUIPMENT STATUS
IS CARD READER READY ?
NO
YES
READ ONE CARD
RELEASE CHANNEL
ADVANCE CARD COUNT IN FILE STATUS
IS 789 PUNCHED IN COLUMN ONE ?
CLEAR CONSOLE MESSAGE
NO
YES
IS 6789 PUNCHED IN COLUMN ONE ?
NO
YES
IS 79 PUNCHED IN COLUMN ONE ?
YES
PROCESS BINARY CARD
NO
WAS LAST RECORD COMPLETE ?
NO
YES
PROCESS HOLLERITH CARD
YES
IS BUFFER EMPTY ?
NO
UPDATE IN ADDRESS IN CENTRAL STORAGE
PAUSE FOR MONITOR
READ RA
IS ERROR FLAG SET ?
NO
YES
RELEASE PPU
SET END OF JOB FLAG
SET END OF RECORD
CLEAR CARD COUNT
EXIT
SET FILE MARK
CLEAR CARD COUNT
EXIT

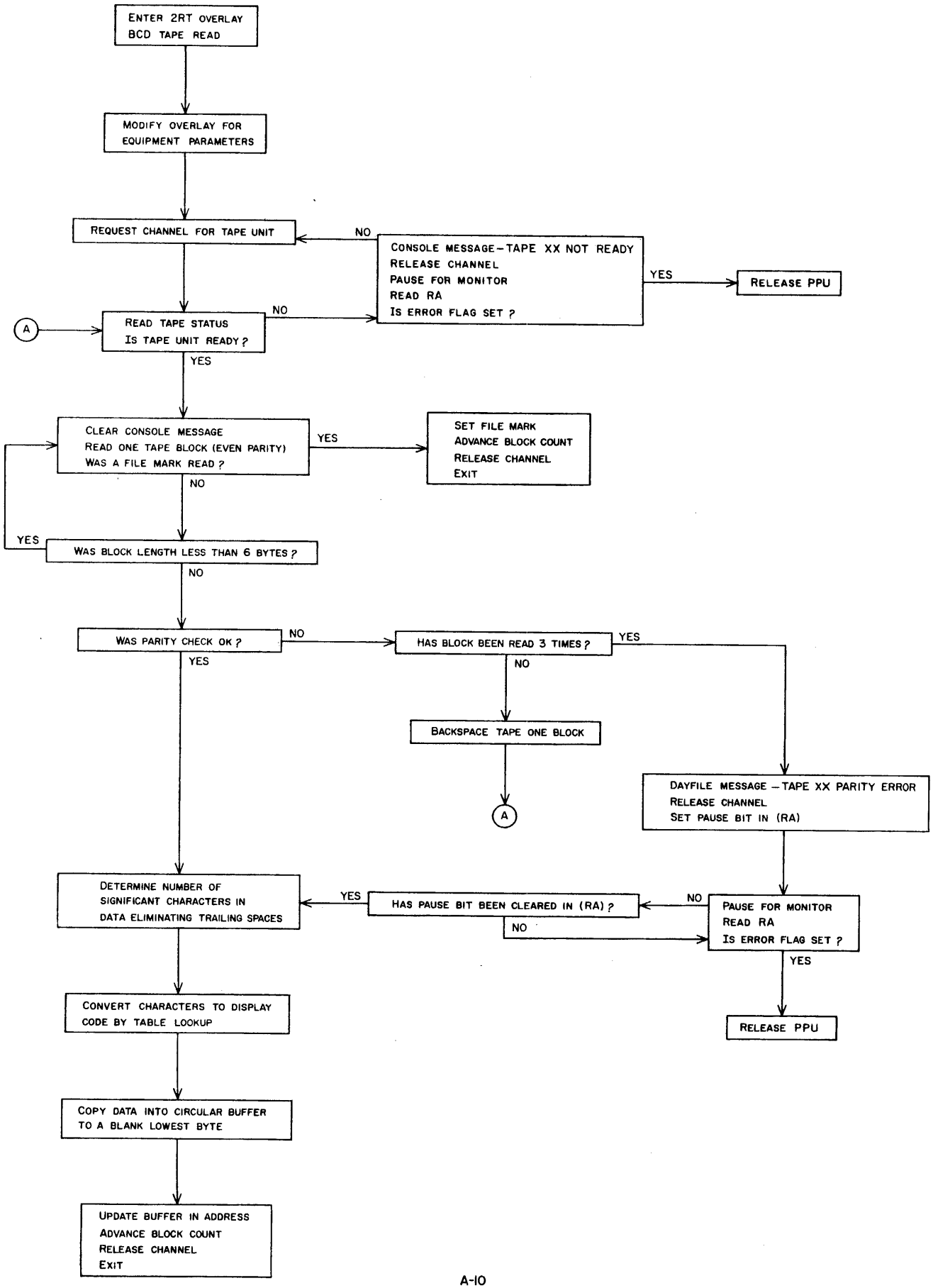


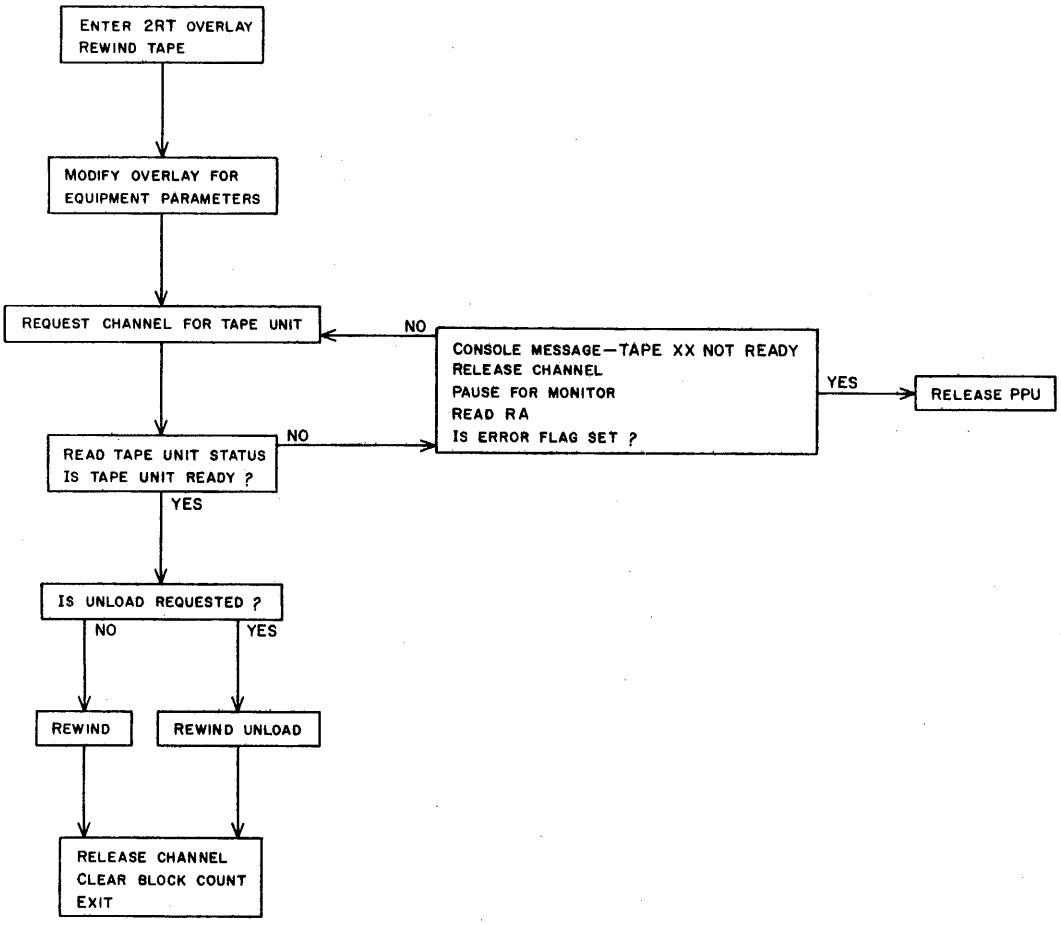
* THIS PATH PROVIDES AN OPPORTUNITY FOR THE OPERATOR TO REREAD THE FAULTY CARD

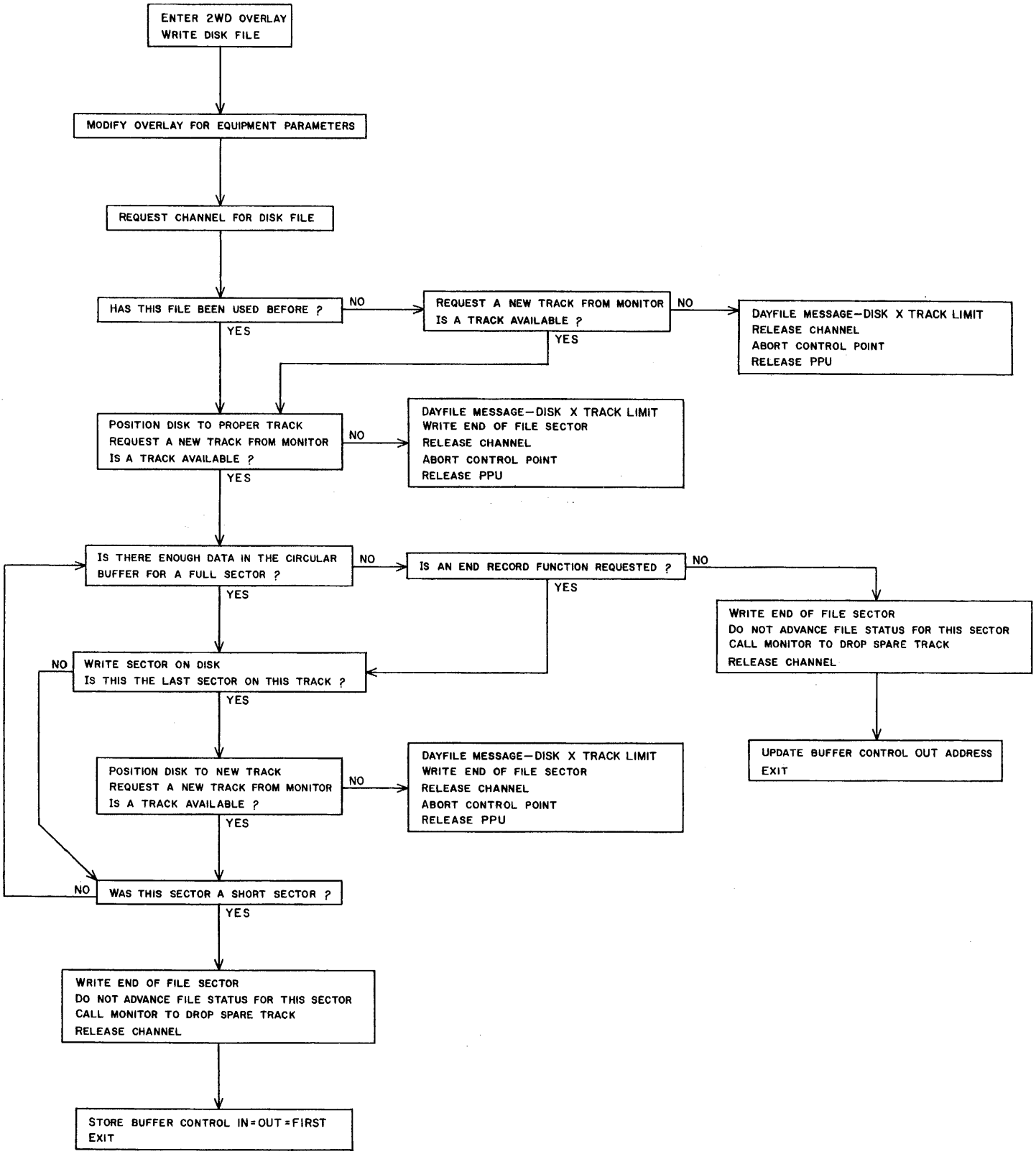

2RC PROCESS HOLLERITH CARD
SENSE TRAILING SPACES AND DETERMINE LAST SIGNIFICANT COLUMN
SET COLUMN INDEX TO FIRST COLUMN
CLEAR WORD BUFFER
SET BYTE INDEX TO FIRST BYTE
CLEAR CHARACTER BUFFER
IS ROW 12 PUNCHED ?
ADD 60B TO CHARACTER
IS ROW 11 PUNCHED ?
ADD 40B TO CHARACTER
IS ROW 0 PUNCHED ?
ADD 20B TO CHARACTER
IS ROW 1 PUNCHED ?
ADD 1 TO CHARACTER
IS ROW 2 PUNCHED ?
ADD 2 TO CHARACTER
IS ROW 3 PUNCHED ?
ADD 3 TO CHARACTER
IS ROW 4 PUNCHED ?
ADD 4 TO CHARACTER
IS ROW 5 PUNCHED ?
ADD 5 TO CHARACTER
IS ROW 6 PUNCHED ?
ADD 6 TO CHARACTER
IS ROW 7 PUNCHED ?
ADD 7 TO CHARACTER
IS ROW 8 PUNCHED ?
ADD 10B TO CHARACTER
IS ROW 9 PUNCHED ?
ADD 11B TO CHARACTER
A
IS THIS FIRST CHARACTER IN BYTE ?
TABLE LOOKUP FOR DISPLAY CODE
STORE CHARACTER IN UPPER HALF OF BYTE
ADVANCE COLUMN INDEX
WAS THIS THE LAST SIGNIFICANT COLUMN ?
STORE WORD IN CIRCULAR BUFFER
ADVANCE CIRCULAR BUFFER IN ADDRESS
DID THIS WORD HAVE DATA IN LAST BYTE ?
STORE CLEARED WORD IN CIRCULAR BUFFER
ADVANCE CIRCULAR BUFFER IN ADDRESS
EXIT
TABLE LOOKUP FOR DISPLAY CODE
ADD CHARACTER IN LOWER HALF OF BYTE
ADVANCE COLUMN INDEX
WAS THIS THE LAST SIGNIFICANT COLUMN ?
ADVANCE BYTE INDEX
WAS THIS THE LAST BYTE IN WORD ?
STORE WORD IN CIRCULAR BUFFER
CLEAR WORD BUFFER
SET BYTE INDEX TO FIRST BYTE
ADVANCE CIRCULAR BUFFER IN ADDRESS
YES
NO

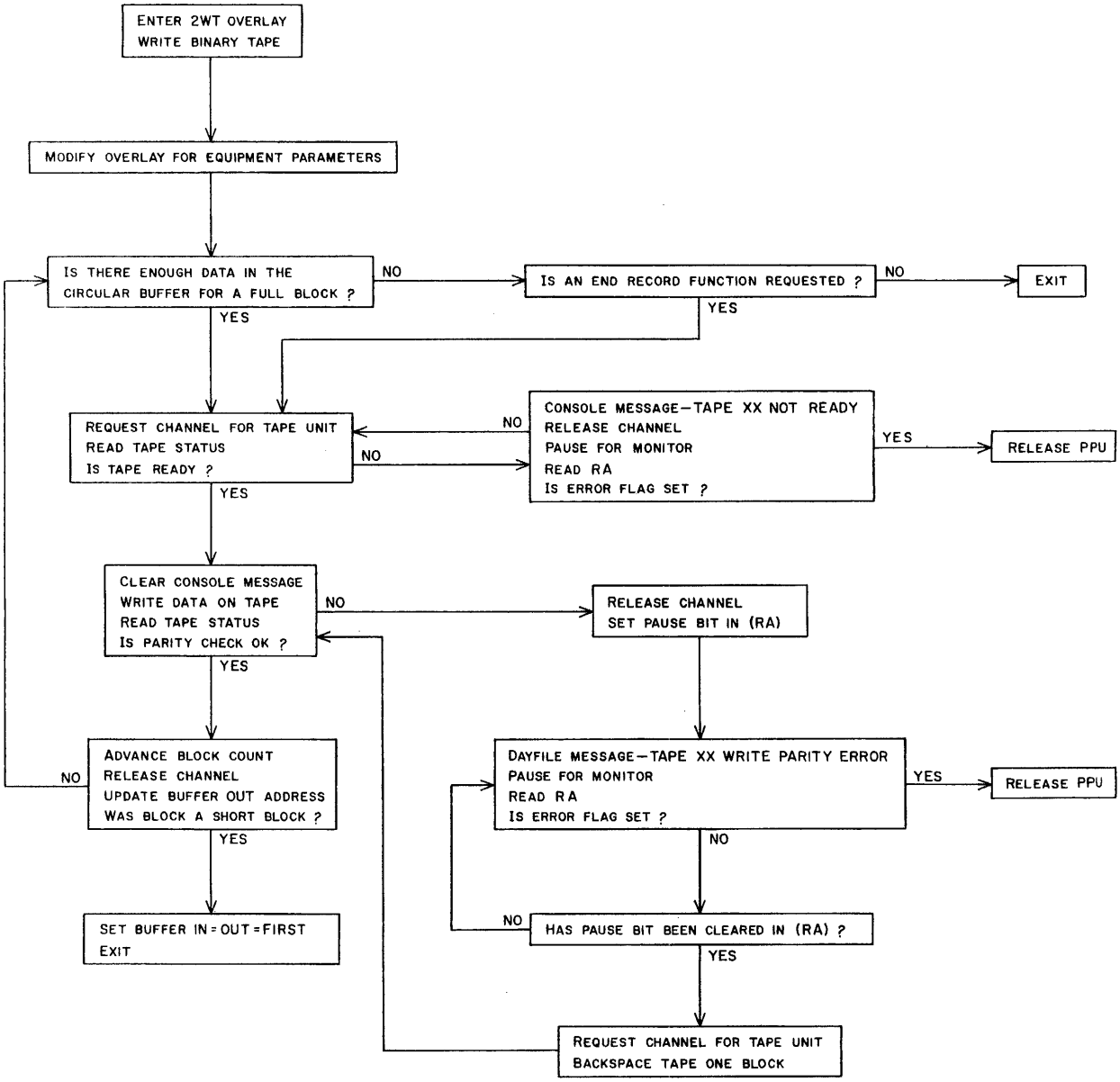

ENTER 2RD OVERLAY
DISK FILE READ
MODIFY OVERLAY FOR
EQUIPMENT PARAMETERS
COMPUTE NUMBER OF SECTORS
WHICH CAN BE LOADED INTO THE
CIRCULAR BUFFER.
IS NUMBER OF SECTORS ZERO ?
YES
EXIT
NO
HAS THIS FILE BEEN USED BEFORE ?
NO
SET FILE MARK
EXIT
YES
REQUEST CHANNEL FOR DISK FILE
POSITION DISK TO PROPER TRACK
READ ONE SECTOR
READ DISK STATUS
IS PARITY OK ?
NO
REREAD SECTOR
READ DISK STATUS
IS PARITY OK ?
NO
SELECT MARGIN 1
REREAD SECTOR
READ DISK STATUS
IS PARITY OK ?
NO
SELECT MARGIN 2
REREAD SECTOR
READ DISK STATUS
IS PARITY OK ?
YES
YES
YES
YES
NO
IS CONTROL BYTE A NEW TRACK NUMBER ?
YES
NO
DAYFILE MESSAGE -
DISK PARITY ERROR
GX TXXX SXXX
STOP
POSITION DISK TO NEW TRACK
ADVANCE FILE STATUS FOR NEXT SECTOR
STORE SECTOR DATA IN CIRCULAR BUFFER
ADVANCE BUFFER IN ADDRESS
IS THIS SECTOR A SHORT SECTOR ?
YES
SET END OF RECORD
RELEASE CHANNEL
NO
YES
IS THERE ROOM FOR ANOTHER SECTOR OF
DATA IN THE CIRCULAR BUFFER ?
NO
RELEASE CHANNEL
UPDATE CIRCULAR BUFFER IN ADDRESS
IS DISK AT FILE MARK ?
YES
NO
SET FILE MARK
EXIT

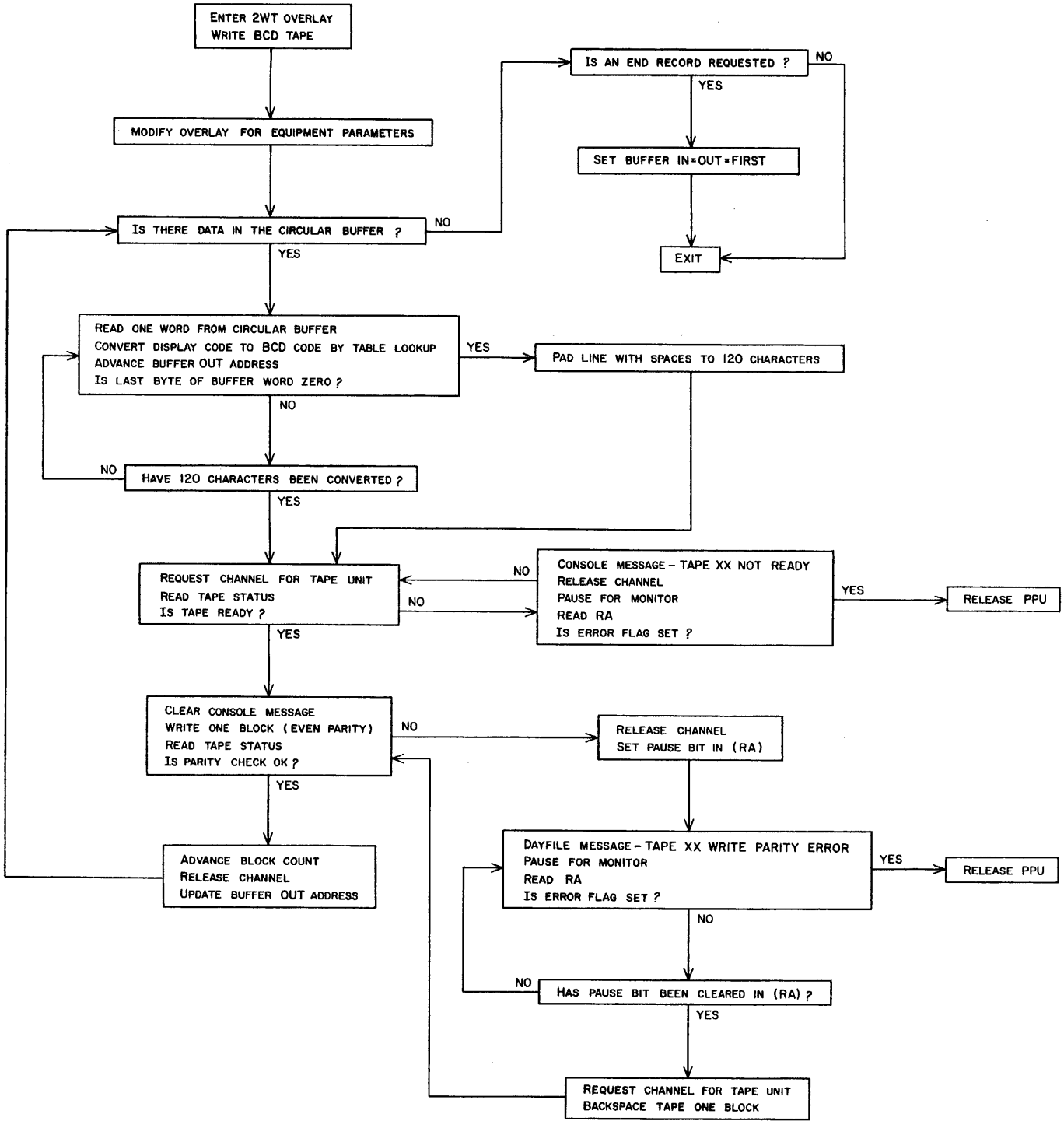

ENTER 2RT OVERLAY
BINARY TAPE READ
MODIFY OVERLAY FOR
EQUIPMENT PARAMETERS
IS THERE ROOM IN THE CIRCULAR
BUFFER FOR A FULL BLOCK OF DATA ?
NO
EXIT
YES
REQUEST CHANNEL FOR TAPE UNIT
NO
CONSOLE MESSAGE—TAPE XX NOT READY
RELEASE CHANNEL
PAUSE FOR MONITOR
READ RA
IS ERROR FLAG SET ?
YES
RELEASE PPU
READ TAPE STATUS
IS TAPE READY ?
NO
YES
CLEAR CONSOLE MESSAGE
READ ONE BLOCK OF TAPE (ODD PARITY)
IS LENGTH AN ODD FRACTION OF WORD ?
NO
YES
YES
IS LENGTH LESS THAN 4 BYTES ?
NO
READ TAPE STATUS
WAS FILE MARK READ ?
YES
SET FILE MARK
ADVANCE BLOCK COUNT
RELEASE CHANNEL
EXIT
NO
IS PARITY CHECK OK ?
YES
ADVANCE BLOCK COUNT
RELEASE CHANNEL
A
NO
HAS BLOCK BEEN READ 3 TIMES ?
YES
DAYFILE MESSAGE —
TAPE XX PARITY ERROR
RELEASE CHANNEL
SET PAUSE BIT IN (RA)
NO
BACKSPACE TAPE ONE BLOCK
PAUSE FOR MONITOR
READ RA
IS ERROR FLAG SET ?
NO
HAS PAUSE BIT BEEN CLEARED IN (RA) ?
YES
NO
YES
RELEASE PPU
A
NO
STORE DATA IN CIRCULAR BUFFER
UPDATE BUFFER IN ADDRESS
WAS BLOCK A SHORT BLOCK ?
YES
SET END OF RECORD
EXIT

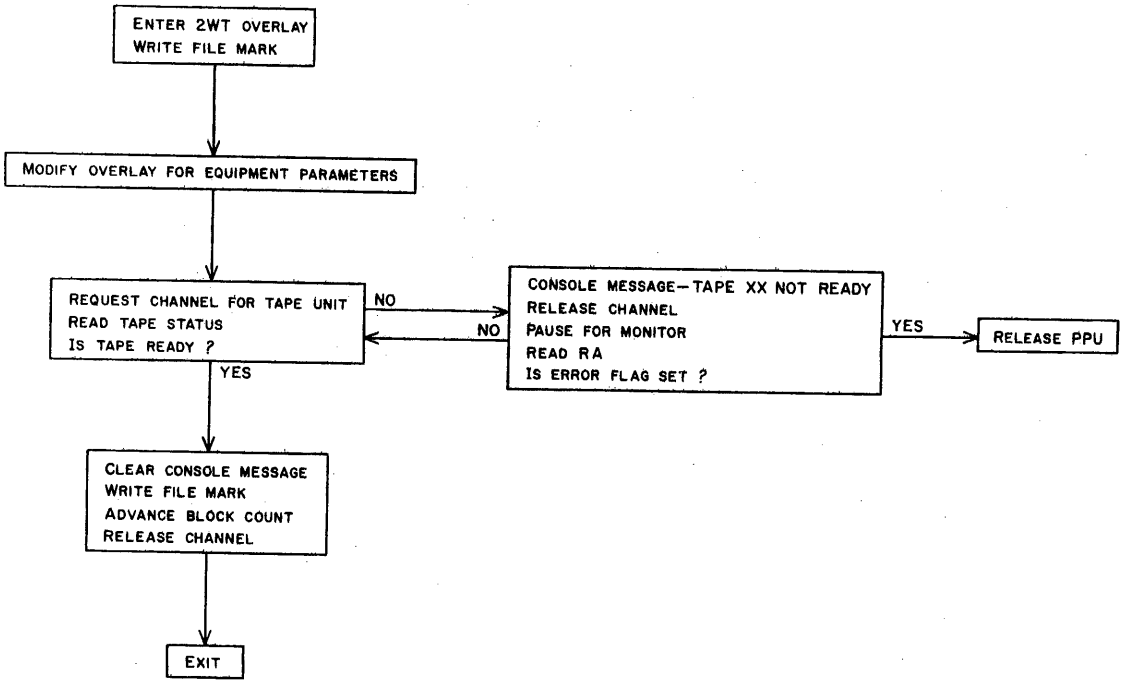

ENTER 2RT OVERLAY
BCD TAPE READ
MODIFY OVERLAY FOR EQUIPMENT PARAMETERS
REQUEST CHANNEL FOR TAPE UNIT
NO
CONSOLE MESSAGE-TAPE XX NOT READY
RELEASE CHANNEL
PAUSE FOR MONITOR
READ RA
IS ERROR FLAG SET ?
YES
RELEASE PPU
A
READ TAPE STATUS
IS TAPE UNIT READY ?
NO
YES
CLEAR CONSOLE MESSAGE
READ ONE TAPE BLOCK (EVEN PARITY)
WAS A FILE MARK READ ?
YES
SET FILE MARK
ADVANCE BLOCK COUNT
RELEASE CHANNEL
EXIT
NO
YES
WAS BLOCK LENGTH LESS THAN 6 BYTES ?
NO
WAS PARITY CHECK OK ?
NO
HAS BLOCK BEEN READ 3 TIMES ?
YES
NO
YES
BACKSPACE TAPE ONE BLOCK
A
DAYFILE MESSAGE - TAPE XX PARITY ERROR
RELEASE CHANNEL
SET PAUSE BIT IN (RA)
DETERMINE NUMBER OF SIGNIFICANT CHARACTERS IN DATA ELIMINATING TRAILING SPACES
YES
HAS PAUSE BIT BEEN CLEARED IN (RA) ?
NO
NO
PAUSE FOR MONITOR
READ RA
IS ERROR FLAG SET ?
YES
RELEASE PPU
CONVERT CHARACTERS TO DISPLAY CODE BY TABLE LOOKUP
COPY DATA INTO CIRCULAR BUFFER TO A BLANK LOWEST BYTE
UPDATE BUFFER IN ADDRESS
ADVANCE BLOCK COUNT
RELEASE CHANNEL
EXIT

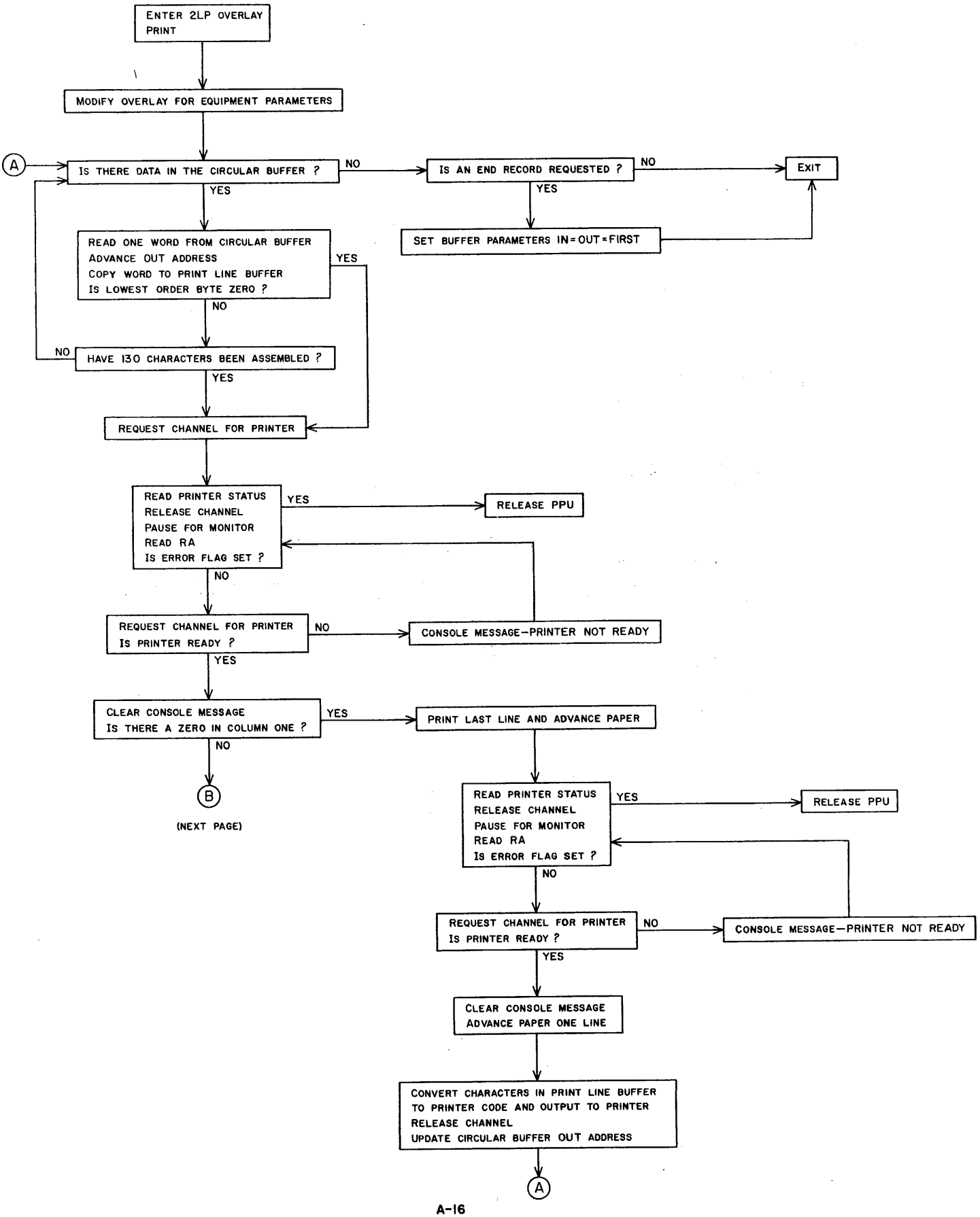

ENTER 2RT OVERLAY
REWIND TAPE
MODIFY OVERLAY FOR
EQUIPMENT PARAMETERS
REQUEST CHANNEL FOR TAPE UNIT
NO
CONSOLE MESSAGE—TAPE XX NOT READY
RELEASE CHANNEL
PAUSE FOR MONITOR
READ RA
IS ERROR FLAG SET ?
YES
RELEASE PPU
READ TAPE UNIT STATUS
IS TAPE UNIT READY ?
NO
YES
IS UNLOAD REQUESTED ?
NO
YES
REWIND
REWIND UNLOAD
RELEASE CHANNEL
CLEAR BLOCK COUNT
EXIT


ENTER 2WD OVERLAY
WRITE DISK FILE
MODIFY OVERLAY FOR EQUIPMENT PARAMETERS
REQUEST CHANNEL FOR DISK FILE
HAS THIS FILE BEEN USED BEFORE ?
NO
YES
REQUEST A NEW TRACK FROM MONITOR
IS A TRACK AVAILABLE ?
NO
YES
DAYFILE MESSAGE—DISK X TRACK LIMIT
RELEASE CHANNEL
ABORT CONTROL POINT
RELEASE PPU
POSITION DISK TO PROPER TRACK
REQUEST A NEW TRACK FROM MONITOR
IS A TRACK AVAILABLE ?
NO
YES
DAYFILE MESSAGE—DISK X TRACK LIMIT
WRITE END OF FILE SECTOR
RELEASE CHANNEL
ABORT CONTROL POINT
RELEASE PPU
IS THERE ENOUGH DATA IN THE CIRCULAR BUFFER FOR A FULL SECTOR ?
NO
YES
IS AN END RECORD FUNCTION REQUESTED ?
NO
YES
WRITE END OF FILE SECTOR
DO NOT ADVANCE FILE STATUS FOR THIS SECTOR
CALL MONITOR TO DROP SPARE TRACK
RELEASE CHANNEL
NO
WRITE SECTOR ON DISK
IS THIS THE LAST SECTOR ON THIS TRACK ?
YES
POSITION DISK TO NEW TRACK
REQUEST A NEW TRACK FROM MONITOR
IS A TRACK AVAILABLE ?
NO
YES
DAYFILE MESSAGE—DISK X TRACK LIMIT
WRITE END OF FILE SECTOR
RELEASE CHANNEL
ABORT CONTROL POINT
RELEASE PPU
UPDATE BUFFER CONTROL OUT ADDRESS
EXIT
NO
WAS THIS SECTOR A SHORT SECTOR ?
YES
WRITE END OF FILE SECTOR
DO NOT ADVANCE FILE STATUS FOR THIS SECTOR
CALL MONITOR TO DROP SPARE TRACK
RELEASE CHANNEL
STORE BUFFER CONTROL IN=OUT=FIRST
EXIT

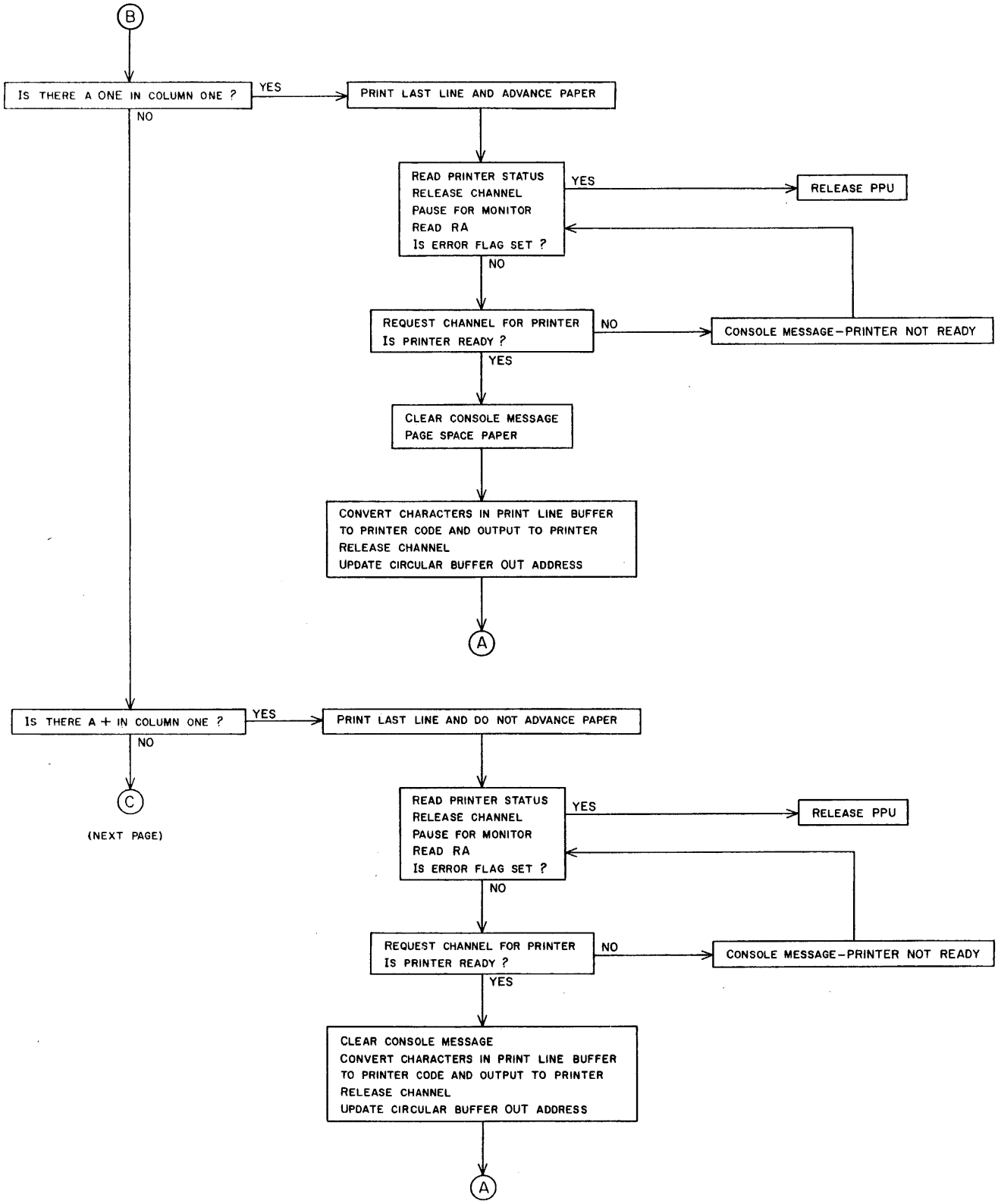

ENTER 2WT OVERLAY
WRITE BINARY TAPE
MODIFY OVERLAY FOR EQUIPMENT PARAMETERS
IS THERE ENOUGH DATA IN THE CIRCULAR BUFFER FOR A FULL BLOCK ?
NO
IS AN END RECORD FUNCTION REQUESTED ?
NO
EXIT
YES
YES
REQUEST CHANNEL FOR TAPE UNIT
READ TAPE STATUS
IS TAPE READY ?
NO
NO
CONSOLE MESSAGE—TAPE XX NOT READY
RELEASE CHANNEL
PAUSE FOR MONITOR
READ RA
IS ERROR FLAG SET ?
YES
RELEASE PPU
YES
CLEAR CONSOLE MESSAGE
WRITE DATA ON TAPE
READ TAPE STATUS
IS PARITY CHECK OK ?
NO
RELEASE CHANNEL
SET PAUSE BIT IN (RA)
YES
ADVANCE BLOCK COUNT
RELEASE CHANNEL
UPDATE BUFFER OUT ADDRESS
WAS BLOCK A SHORT BLOCK ?
NO
YES
DAYFILE MESSAGE—TAPE XX WRITE PARITY ERROR
PAUSE FOR MONITOR
READ RA
IS ERROR FLAG SET ?
YES
RELEASE PPU
NO
SET BUFFER IN = OUT = FIRST
EXIT
NO
HAS PAUSE BIT BEEN CLEARED IN (RA) ?
YES
REQUEST CHANNEL FOR TAPE UNIT
BACKSPACE TAPE ONE BLOCK


ENTER 2WT OVERLAY
WRITE BCD TAPE
MODIFY OVERLAY FOR EQUIPMENT PARAMETERS
IS THERE DATA IN THE CIRCULAR BUFFER ?
NO
YES
IS AN END RECORD REQUESTED ?
NO
YES
SET BUFFER IN=OUT=FIRST
EXIT
READ ONE WORD FROM CIRCULAR BUFFER
CONVERT DISPLAY CODE TO BCD CODE BY TABLE LOOKUP
ADVANCE BUFFER OUT ADDRESS
IS LAST BYTE OF BUFFER WORD ZERO ?
YES
PAD LINE WITH SPACES TO 120 CHARACTERS
NO
NO
HAVE 120 CHARACTERS BEEN CONVERTED ?
YES
REQUEST CHANNEL FOR TAPE UNIT
READ TAPE STATUS
IS TAPE READY ?
NO
NO
CONSOLE MESSAGE - TAPE XX NOT READY
RELEASE CHANNEL
PAUSE FOR MONITOR
READ RA
IS ERROR FLAG SET ?
YES
RELEASE PPU
YES
CLEAR CONSOLE MESSAGE
WRITE ONE BLOCK (EVEN PARITY)
READ TAPE STATUS
IS PARITY CHECK OK ?
NO
RELEASE CHANNEL
SET PAUSE BIT IN (RA)
YES
ADVANCE BLOCK COUNT
RELEASE CHANNEL
UPDATE BUFFER OUT ADDRESS
DAYFILE MESSAGE - TAPE XX WRITE PARITY ERROR
PAUSE FOR MONITOR
READ RA
IS ERROR FLAG SET ?
YES
RELEASE PPU
NO
NO
HAS PAUSE BIT BEEN CLEARED IN (RA) ?
YES
REQUEST CHANNEL FOR TAPE UNIT
BACKSPACE TAPE ONE BLOCK

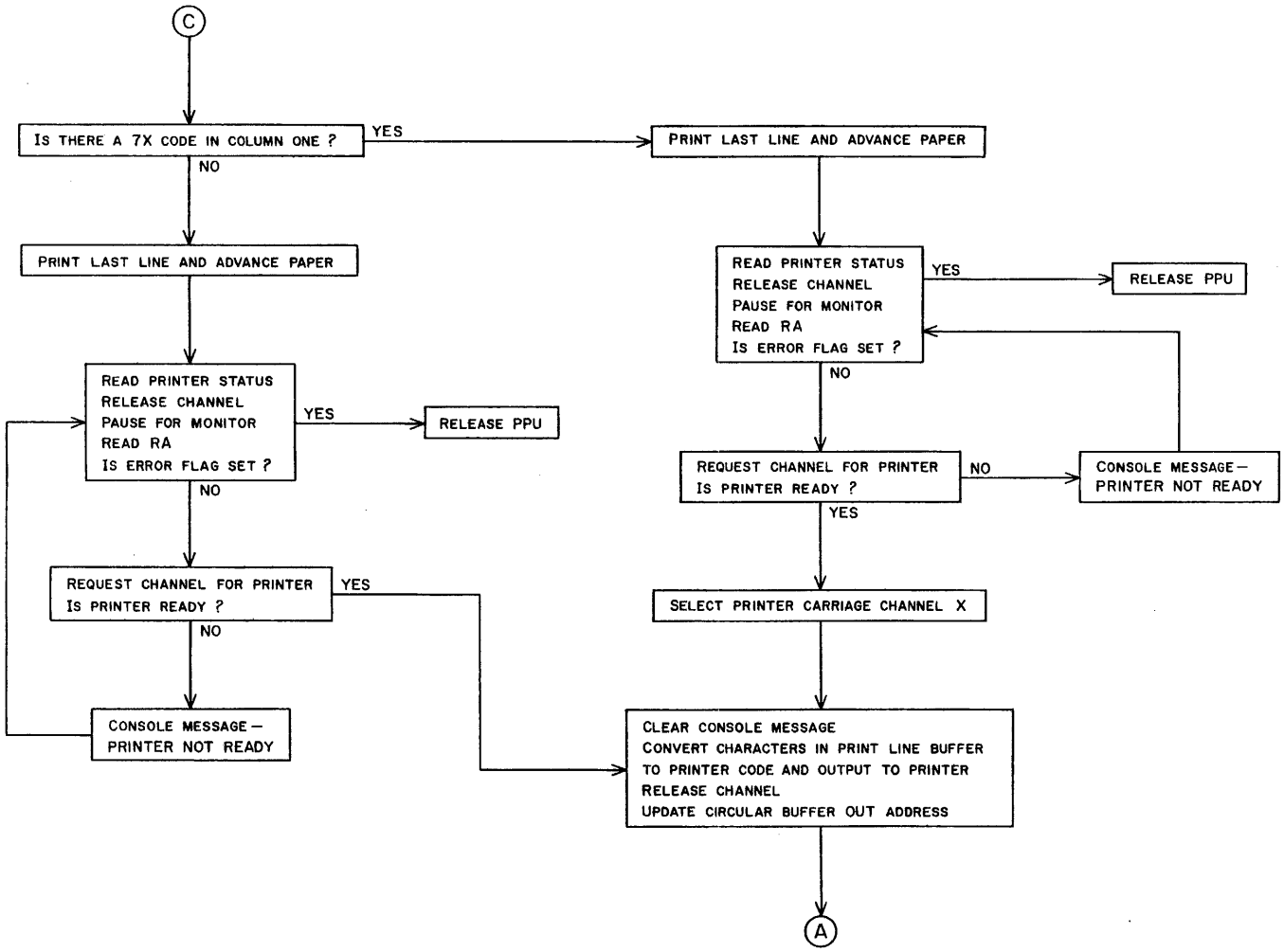

ENTER 2WT OVERLAY
WRITE FILE MARK
MODIFY OVERLAY FOR EQUIPMENT PARAMETERS
REQUEST CHANNEL FOR TAPE UNIT
READ TAPE STATUS
IS TAPE READY ?
NO
NO
YES
CONSOLE MESSAGE—TAPE XX NOT READY
RELEASE CHANNEL
PAUSE FOR MONITOR
READ RA
IS ERROR FLAG SET ?
YES
RELEASE PPU
CLEAR CONSOLE MESSAGE
WRITE FILE MARK
ADVANCE BLOCK COUNT
RELEASE CHANNEL
EXIT

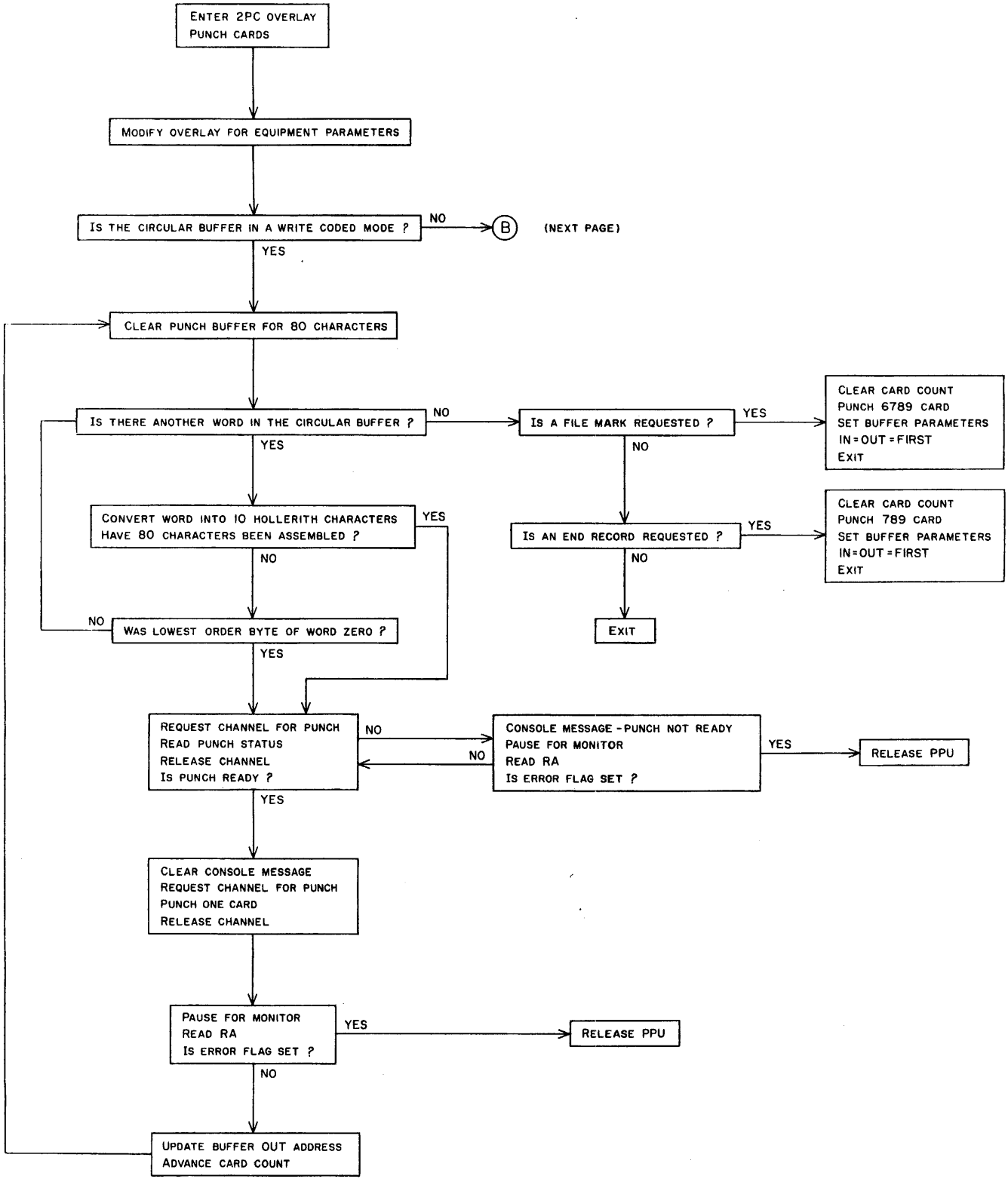

ENTER 2LP OVERLAY PRINT
MODIFY OVERLAY FOR EQUIPMENT PARAMETERS
A
IS THERE DATA IN THE CIRCULAR BUFFER ?
NO
IS AN END RECORD REQUESTED ?
NO
EXIT
YES
YES
SET BUFFER PARAMETERS IN=OUT=FIRST
READ ONE WORD FROM CIRCULAR BUFFER
ADVANCE OUT ADDRESS
COPY WORD TO PRINT LINE BUFFER
IS LOWEST ORDER BYTE ZERO ?
YES
NO
NO
HAVE 130 CHARACTERS BEEN ASSEMBLED ?
YES
REQUEST CHANNEL FOR PRINTER
READ PRINTER STATUS
RELEASE CHANNEL
PAUSE FOR MONITOR
READ RA
IS ERROR FLAG SET ?
YES
RELEASE PPU
NO
REQUEST CHANNEL FOR PRINTER
IS PRINTER READY ?
NO
CONSOLE MESSAGE—PRINTER NOT READY
YES
CLEAR CONSOLE MESSAGE
IS THERE A ZERO IN COLUMN ONE ?
YES
PRINT LAST LINE AND ADVANCE PAPER
NO
B
(NEXT PAGE)
READ PRINTER STATUS
RELEASE CHANNEL
PAUSE FOR MONITOR
READ RA
IS ERROR FLAG SET ?
YES
RELEASE PPU
NO
REQUEST CHANNEL FOR PRINTER
IS PRINTER READY ?
NO
CONSOLE MESSAGE—PRINTER NOT READY
YES
CLEAR CONSOLE MESSAGE
ADVANCE PAPER ONE LINE
CONVERT CHARACTERS IN PRINT LINE BUFFER
TO PRINTER CODE AND OUTPUT TO PRINTER
RELEASE CHANNEL
UPDATE CIRCULAR BUFFER OUT ADDRESS
A

(2LP CONTINUED)

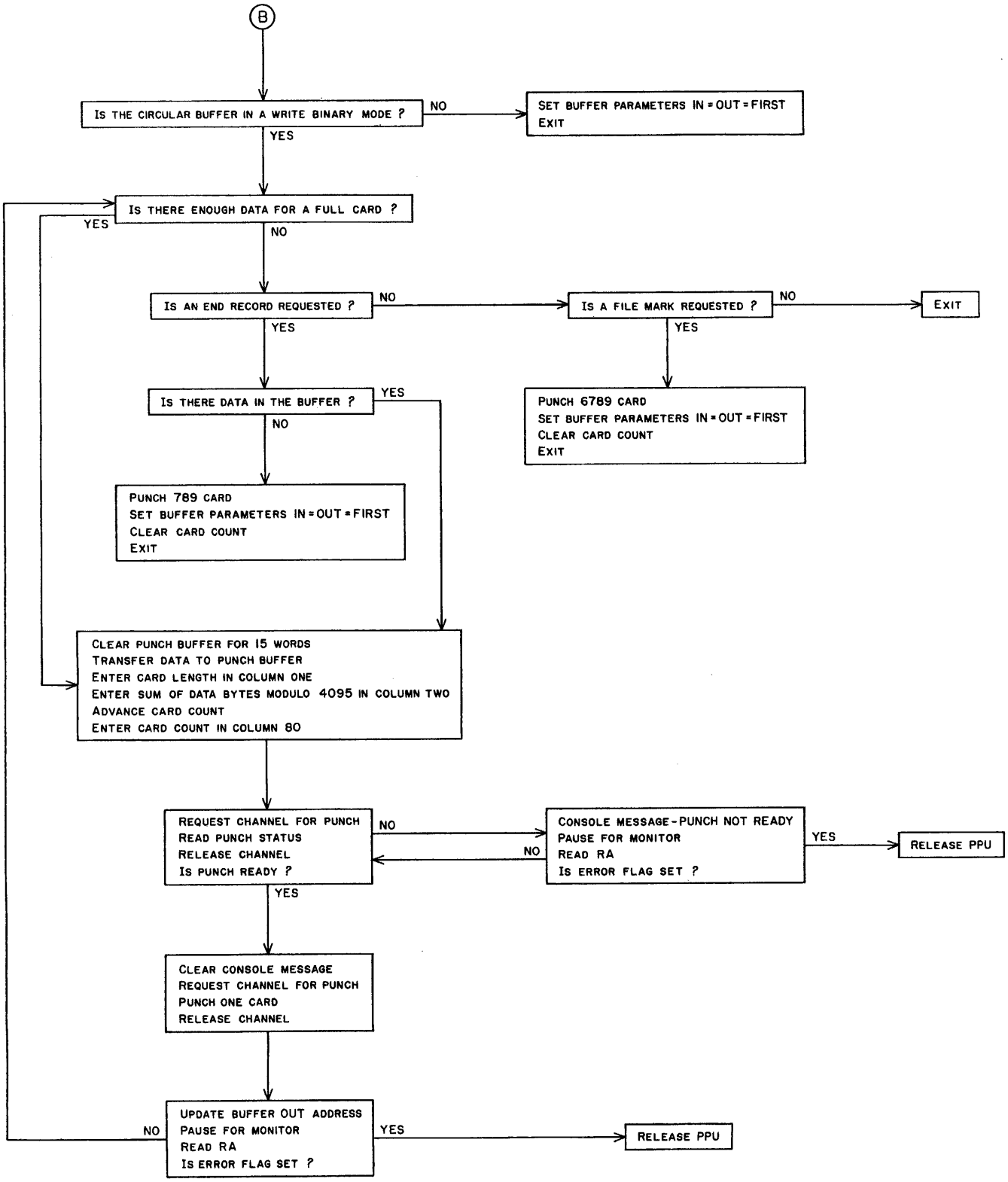


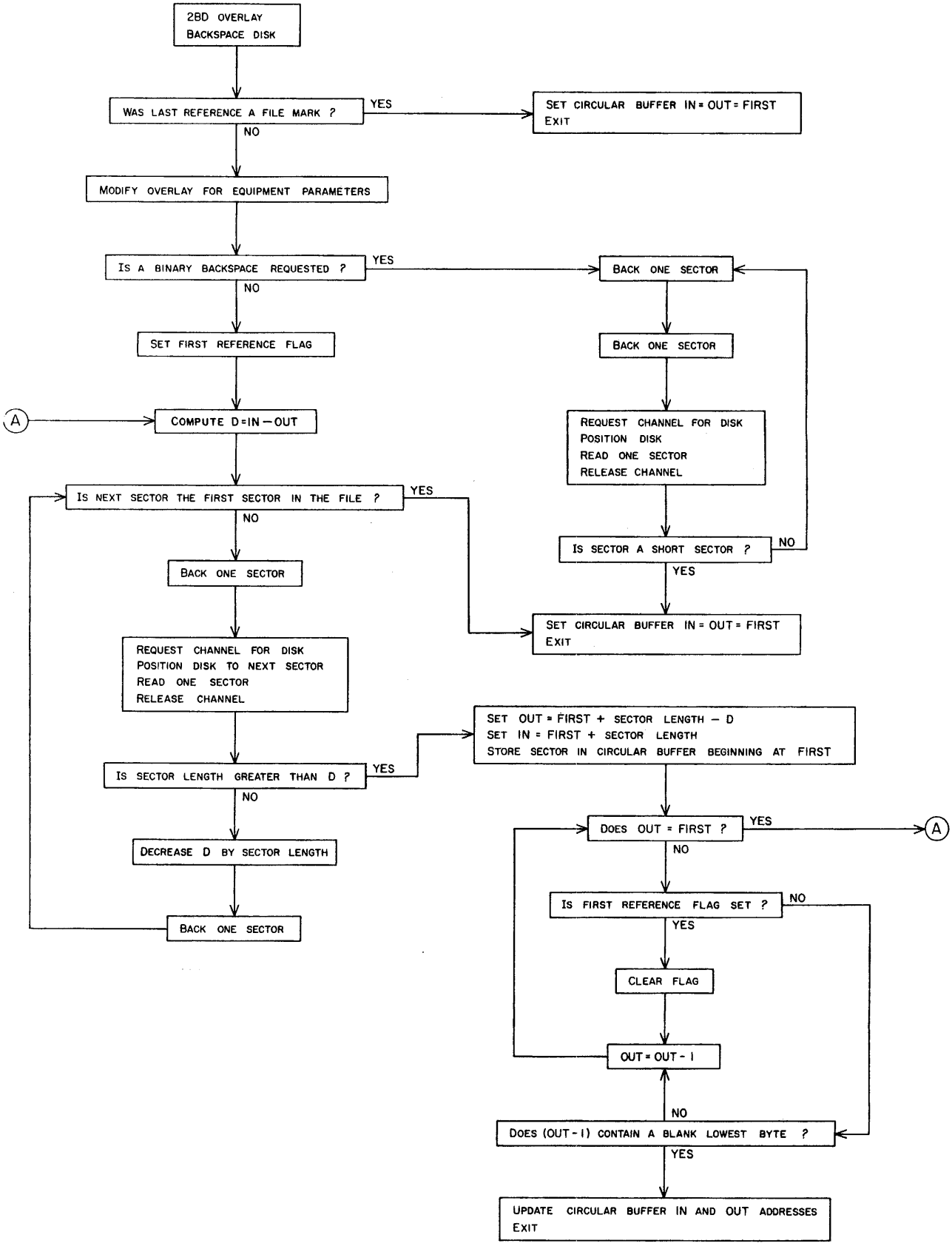
(2LP CONTINUED)

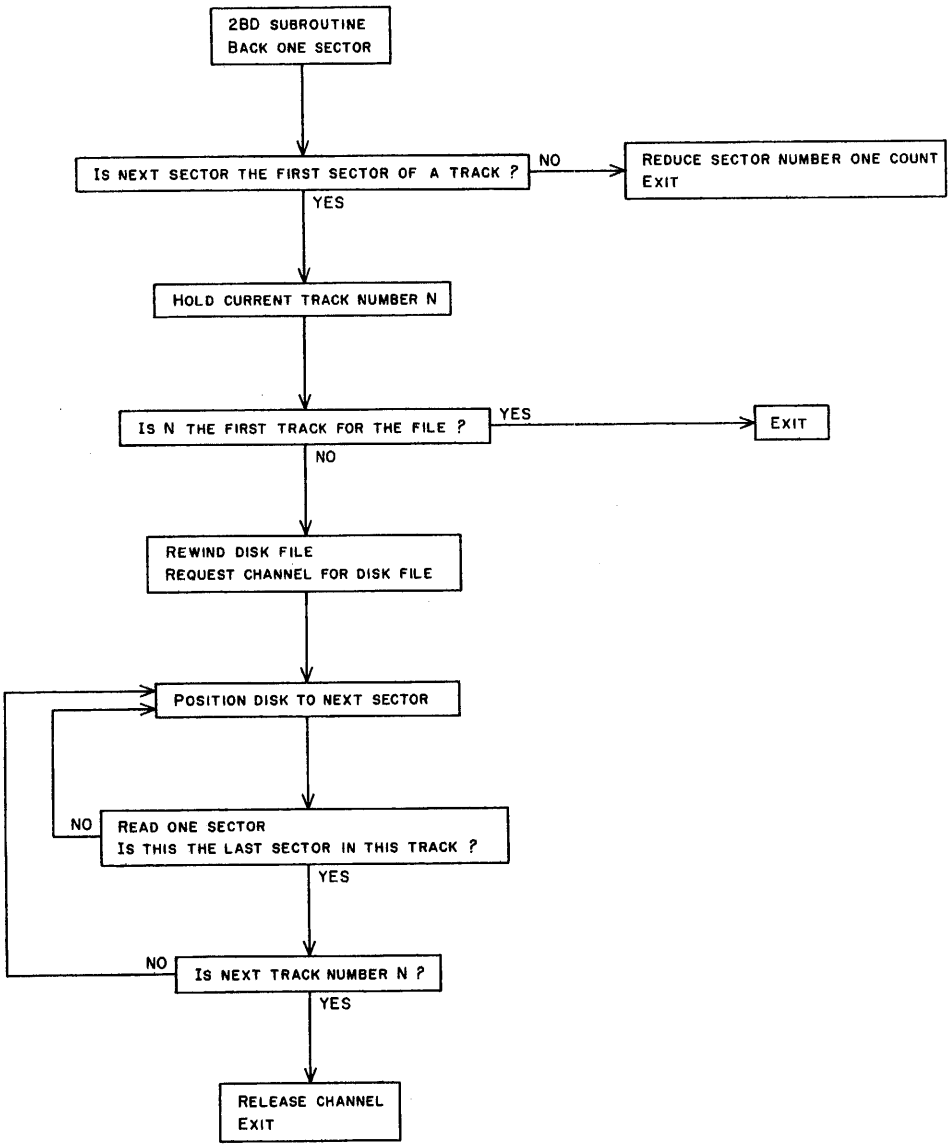


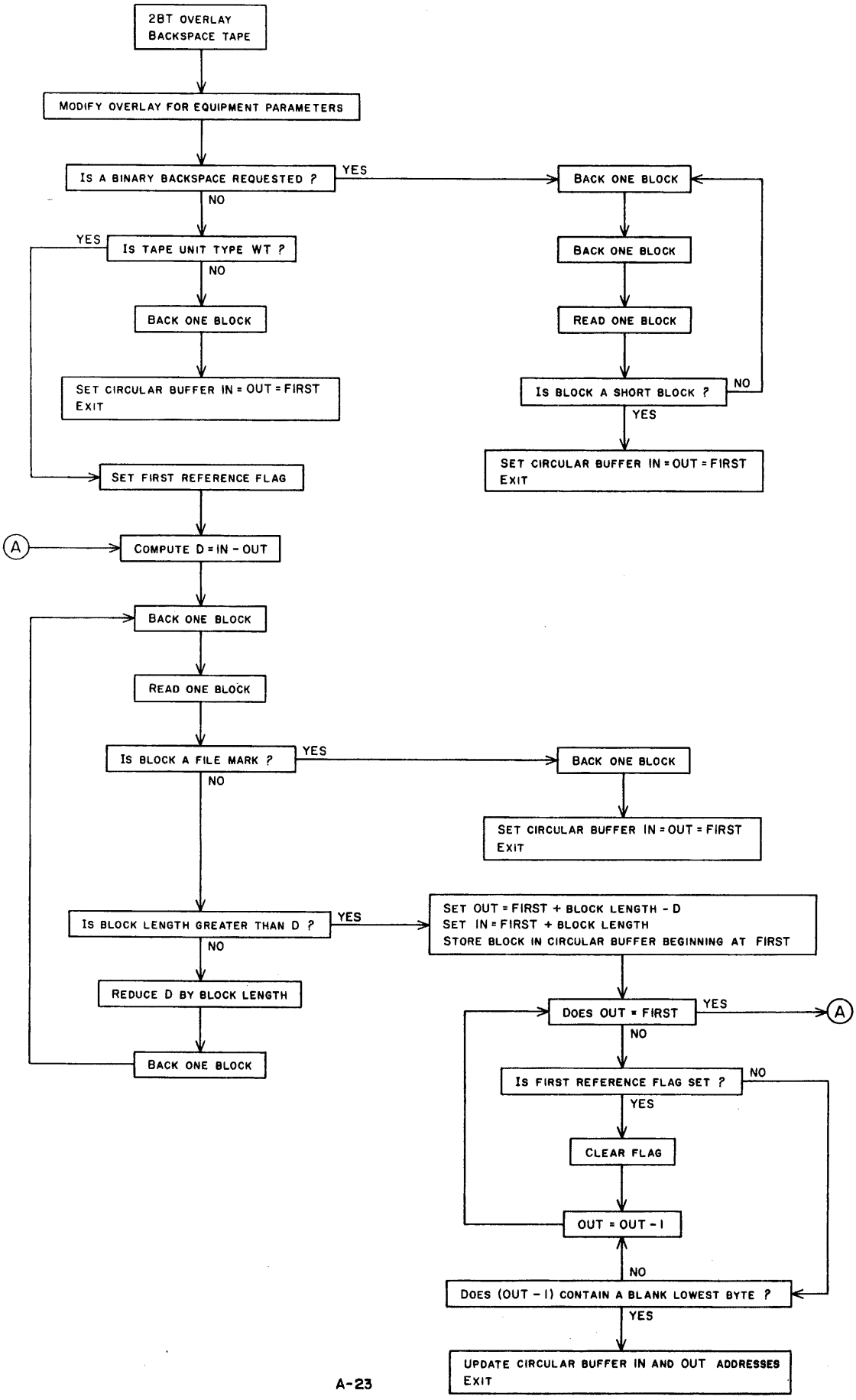

ENTER 2PC OVERLAY
PUNCH CARDS
MODIFY OVERLAY FOR EQUIPMENT PARAMETERS
IS THE CIRCULAR BUFFER IN A WRITE CODED MODE ?
NO
B
(NEXT PAGE)
YES
CLEAR PUNCH BUFFER FOR 80 CHARACTERS
IS THERE ANOTHER WORD IN THE CIRCULAR BUFFER ?
NO
IS A FILE MARK REQUESTED ?
YES
CLEAR CARD COUNT
PUNCH 6789 CARD
SET BUFFER PARAMETERS
IN=OUT=FIRST
EXIT
NO
YES
CONVERT WORD INTO 10 HOLLERITH CHARACTERS
HAVE 80 CHARACTERS BEEN ASSEMBLED ?
YES
IS AN END RECORD REQUESTED ?
YES
CLEAR CARD COUNT
PUNCH 789 CARD
SET BUFFER PARAMETERS
IN=OUT=FIRST
EXIT
NO
NO
EXIT
NO
WAS LOWEST ORDER BYTE OF WORD ZERO ?
YES
REQUEST CHANNEL FOR PUNCH
READ PUNCH STATUS
RELEASE CHANNEL
IS PUNCH READY ?
NO
CONSOLE MESSAGE - PUNCH NOT READY
PAUSE FOR MONITOR
READ RA
IS ERROR FLAG SET ?
YES
RELEASE PPU
NO
YES
CLEAR CONSOLE MESSAGE
REQUEST CHANNEL FOR PUNCH
PUNCH ONE CARD
RELEASE CHANNEL
PAUSE FOR MONITOR
READ RA
IS ERROR FLAG SET ?
YES
RELEASE PPU
NO
UPDATE BUFFER OUT ADDRESS
ADVANCE CARD COUNT

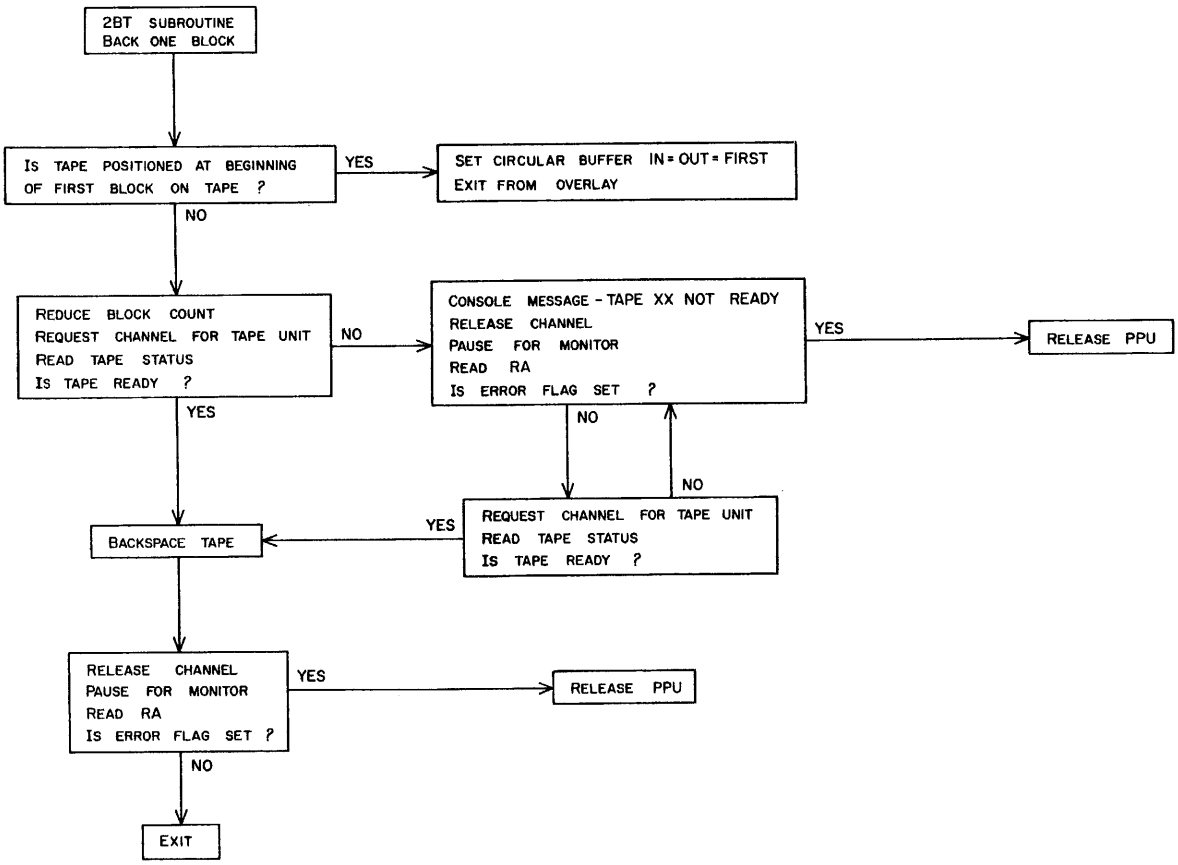
(2PC OVERLAY CONTINUED)

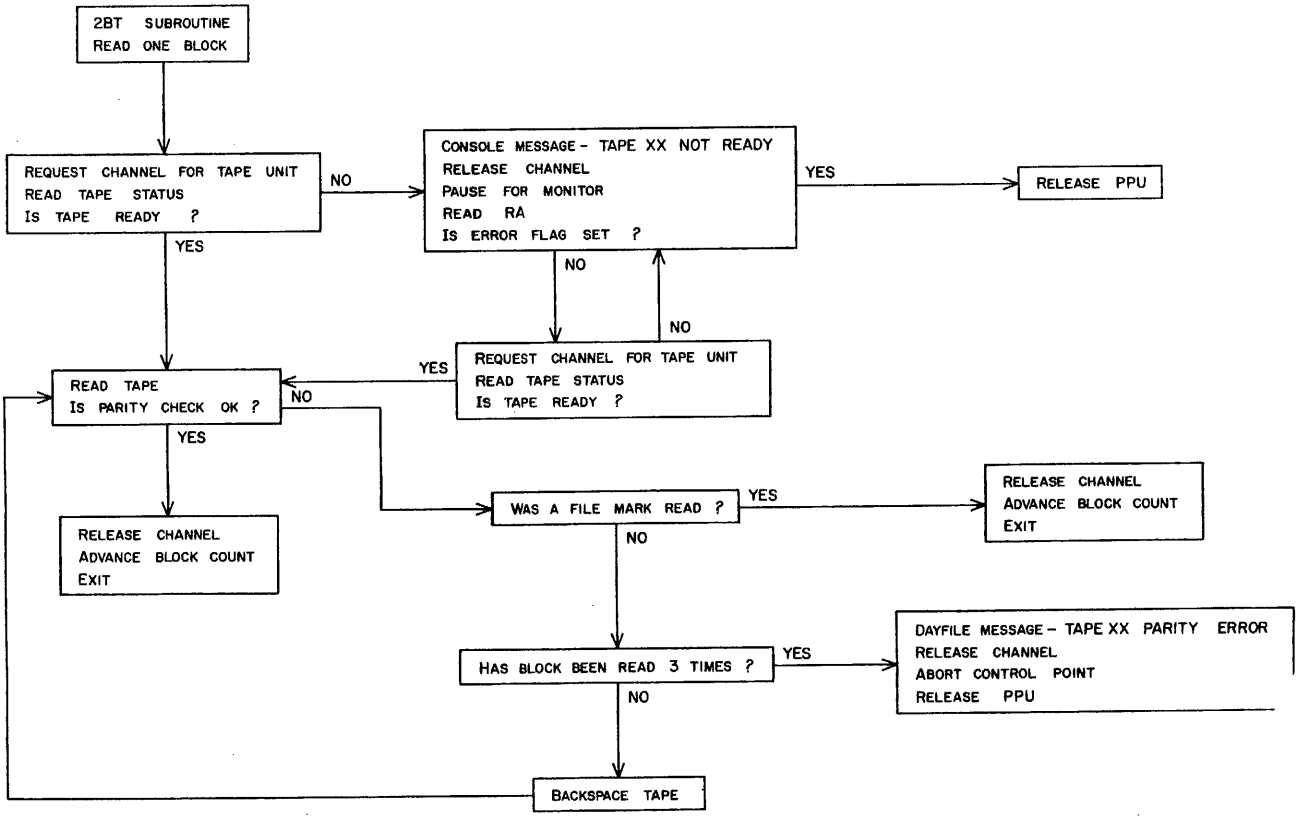


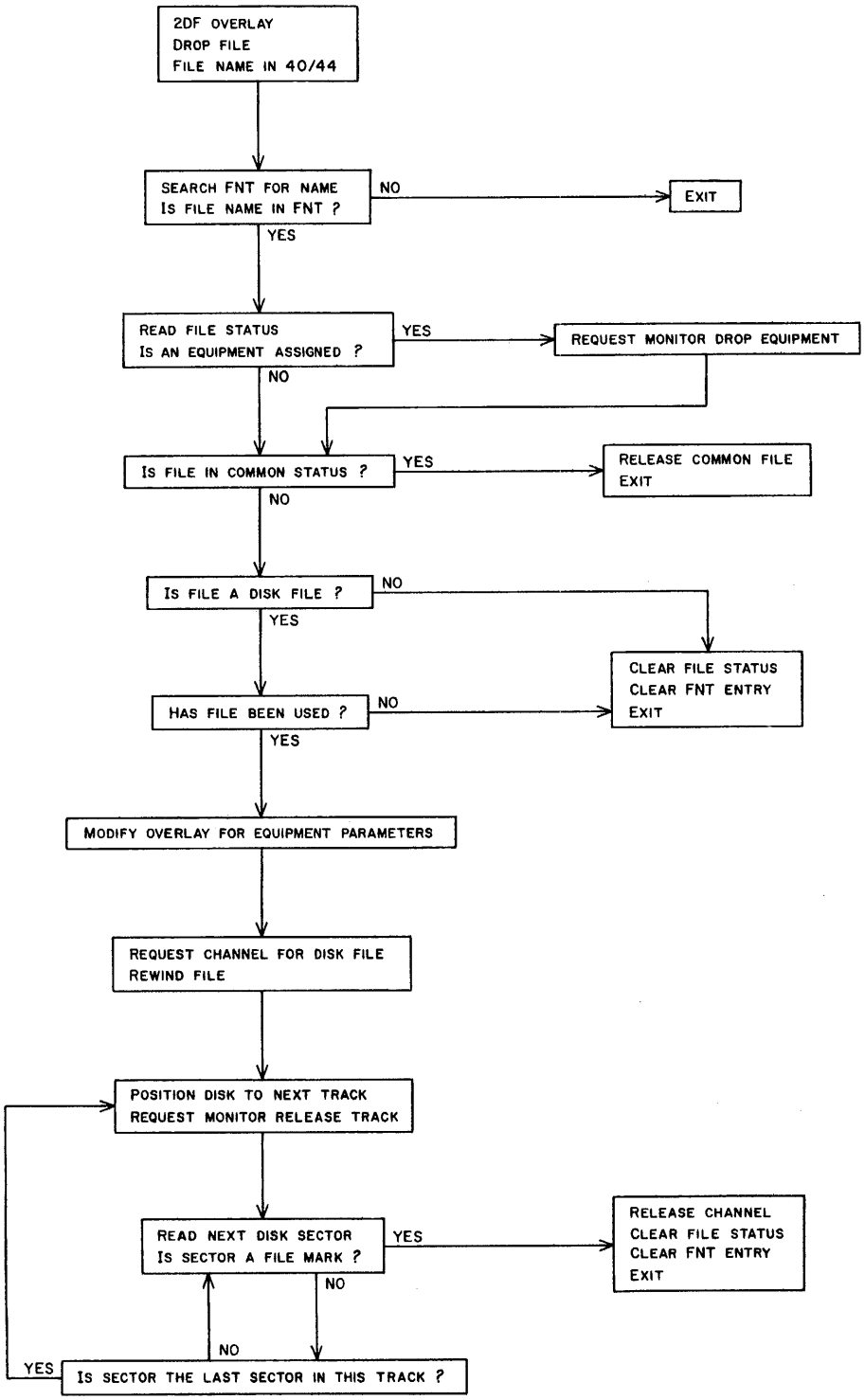

2BD OVERLAY
BACKSPACE DISK
WAS LAST REFERENCE A FILE MARK ?
YES
SET CIRCULAR BUFFER IN = OUT = FIRST
EXIT
NO
MODIFY OVERLAY FOR EQUIPMENT PARAMETERS
IS A BINARY BACKSPACE REQUESTED ?
YES
BACK ONE SECTOR
NO
SET FIRST REFERENCE FLAG
BACK ONE SECTOR
A
COMPUTE D=IN−OUT
REQUEST CHANNEL FOR DISK
POSITION DISK
READ ONE SECTOR
RELEASE CHANNEL
IS NEXT SECTOR THE FIRST SECTOR IN THE FILE ?
YES
NO
IS SECTOR A SHORT SECTOR ?
NO
YES
BACK ONE SECTOR
SET CIRCULAR BUFFER IN = OUT = FIRST
EXIT
REQUEST CHANNEL FOR DISK
POSITION DISK TO NEXT SECTOR
READ ONE SECTOR
RELEASE CHANNEL
SET OUT = FIRST + SECTOR LENGTH − D
SET IN = FIRST + SECTOR LENGTH
STORE SECTOR IN CIRCULAR BUFFER BEGINNING AT FIRST
IS SECTOR LENGTH GREATER THAN D ?
YES
NO
DOES OUT = FIRST ?
YES
A
NO
DECREASE D BY SECTOR LENGTH
IS FIRST REFERENCE FLAG SET ?
NO
YES
CLEAR FLAG
OUT = OUT - 1
BACK ONE SECTOR
NO
DOES (OUT - 1) CONTAIN A BLANK LOWEST BYTE ?
YES
UPDATE CIRCULAR BUFFER IN AND OUT ADDRESSES
EXIT

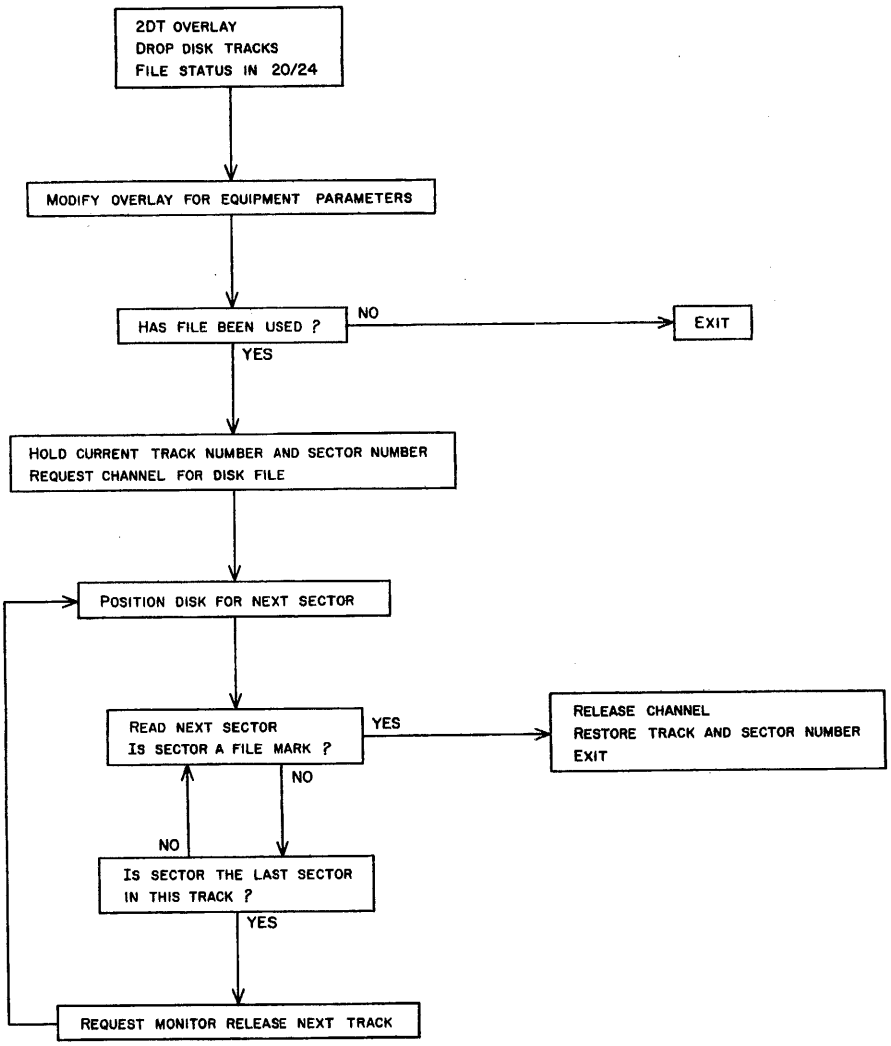

2BD SUBROUTINE
BACK ONE SECTOR
IS NEXT SECTOR THE FIRST SECTOR OF A TRACK ?
NO
REDUCE SECTOR NUMBER ONE COUNT
EXIT
YES
HOLD CURRENT TRACK NUMBER N
IS N THE FIRST TRACK FOR THE FILE ?
YES
EXIT
NO
REWIND DISK FILE
REQUEST CHANNEL FOR DISK FILE
POSITION DISK TO NEXT SECTOR
NO
READ ONE SECTOR
IS THIS THE LAST SECTOR IN THIS TRACK ?
YES
NO
IS NEXT TRACK NUMBER N ?
YES
RELEASE CHANNEL
EXIT

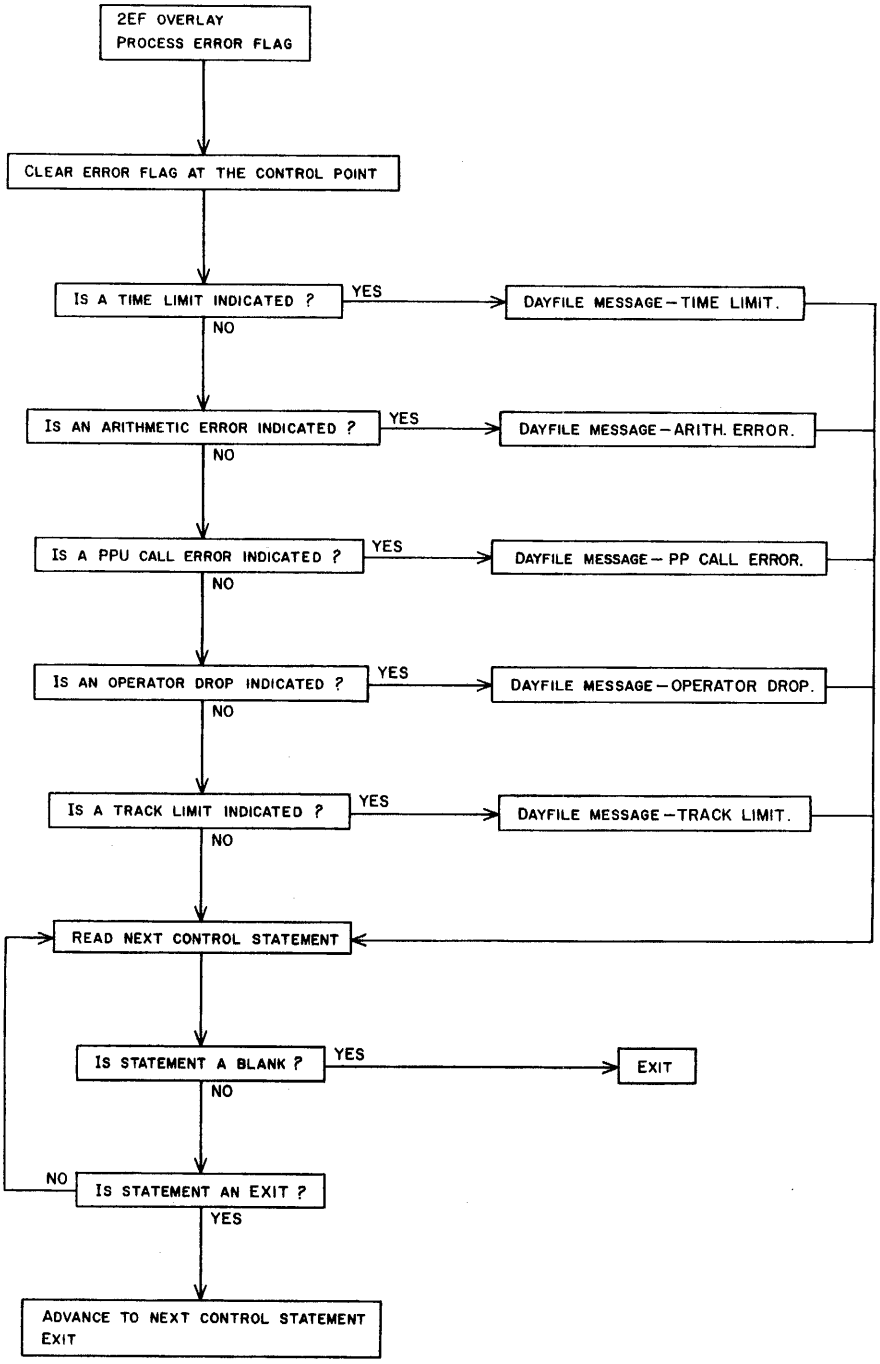

2BT OVERLAY
BACKSPACE TAPE
MODIFY OVERLAY FOR EQUIPMENT PARAMETERS
IS A BINARY BACKSPACE REQUESTED ?
YES
NO
BACK ONE BLOCK
BACK ONE BLOCK
READ ONE BLOCK
IS BLOCK A SHORT BLOCK ?
NO
YES
SET CIRCULAR BUFFER IN = OUT = FIRST
EXIT
IS TAPE UNIT TYPE WT ?
YES
NO
BACK ONE BLOCK
SET CIRCULAR BUFFER IN = OUT = FIRST
EXIT
SET FIRST REFERENCE FLAG
A
COMPUTE D = IN - OUT
BACK ONE BLOCK
READ ONE BLOCK
IS BLOCK A FILE MARK ?
YES
NO
BACK ONE BLOCK
SET CIRCULAR BUFFER IN = OUT = FIRST
EXIT
IS BLOCK LENGTH GREATER THAN D ?
YES
NO
SET OUT = FIRST + BLOCK LENGTH - D
SET IN = FIRST + BLOCK LENGTH
STORE BLOCK IN CIRCULAR BUFFER BEGINNING AT FIRST
REDUCE D BY BLOCK LENGTH
BACK ONE BLOCK
DOES OUT = FIRST
YES
A
NO
IS FIRST REFERENCE FLAG SET ?
NO
YES
CLEAR FLAG
OUT = OUT - 1
NO
DOES (OUT - 1) CONTAIN A BLANK LOWEST BYTE ?
YES
UPDATE CIRCULAR BUFFER IN AND OUT ADDRESSES
EXIT

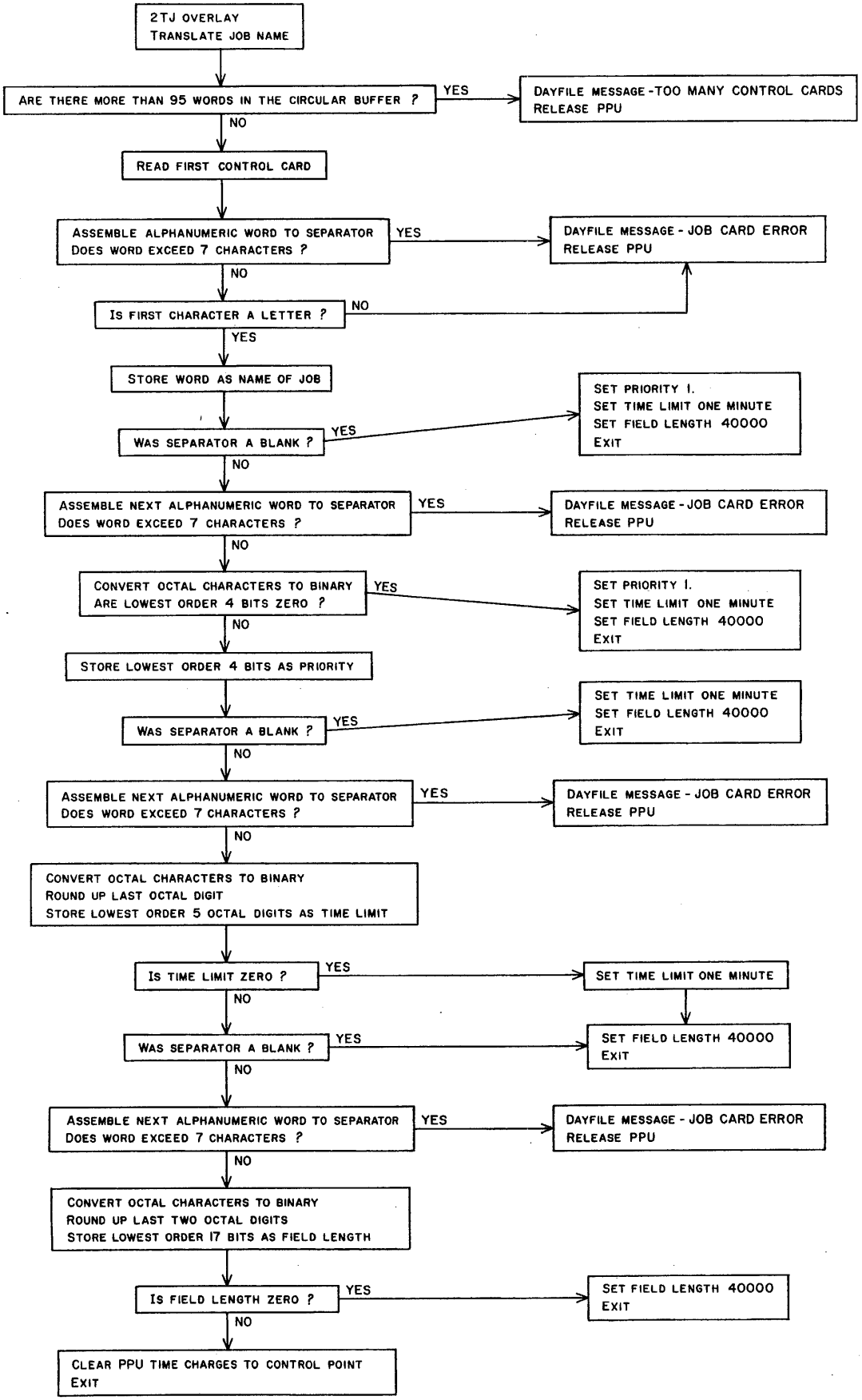

2BT SUBROUTINE
BACK ONE BLOCK
IS TAPE POSITIONED AT BEGINNING OF FIRST BLOCK ON TAPE ?
YES
SET CIRCULAR BUFFER IN = OUT = FIRST
EXIT FROM OVERLAY
NO
REDUCE BLOCK COUNT
REQUEST CHANNEL FOR TAPE UNIT
READ TAPE STATUS
IS TAPE READY ?
NO
CONSOLE MESSAGE - TAPE XX NOT READY
RELEASE CHANNEL
PAUSE FOR MONITOR
READ RA
IS ERROR FLAG SET ?
YES
RELEASE PPU
YES
NO
NO
BACKSPACE TAPE
YES
REQUEST CHANNEL FOR TAPE UNIT
READ TAPE STATUS
IS TAPE READY ?
RELEASE CHANNEL
PAUSE FOR MONITOR
READ RA
IS ERROR FLAG SET ?
YES
RELEASE PPU
NO
EXIT

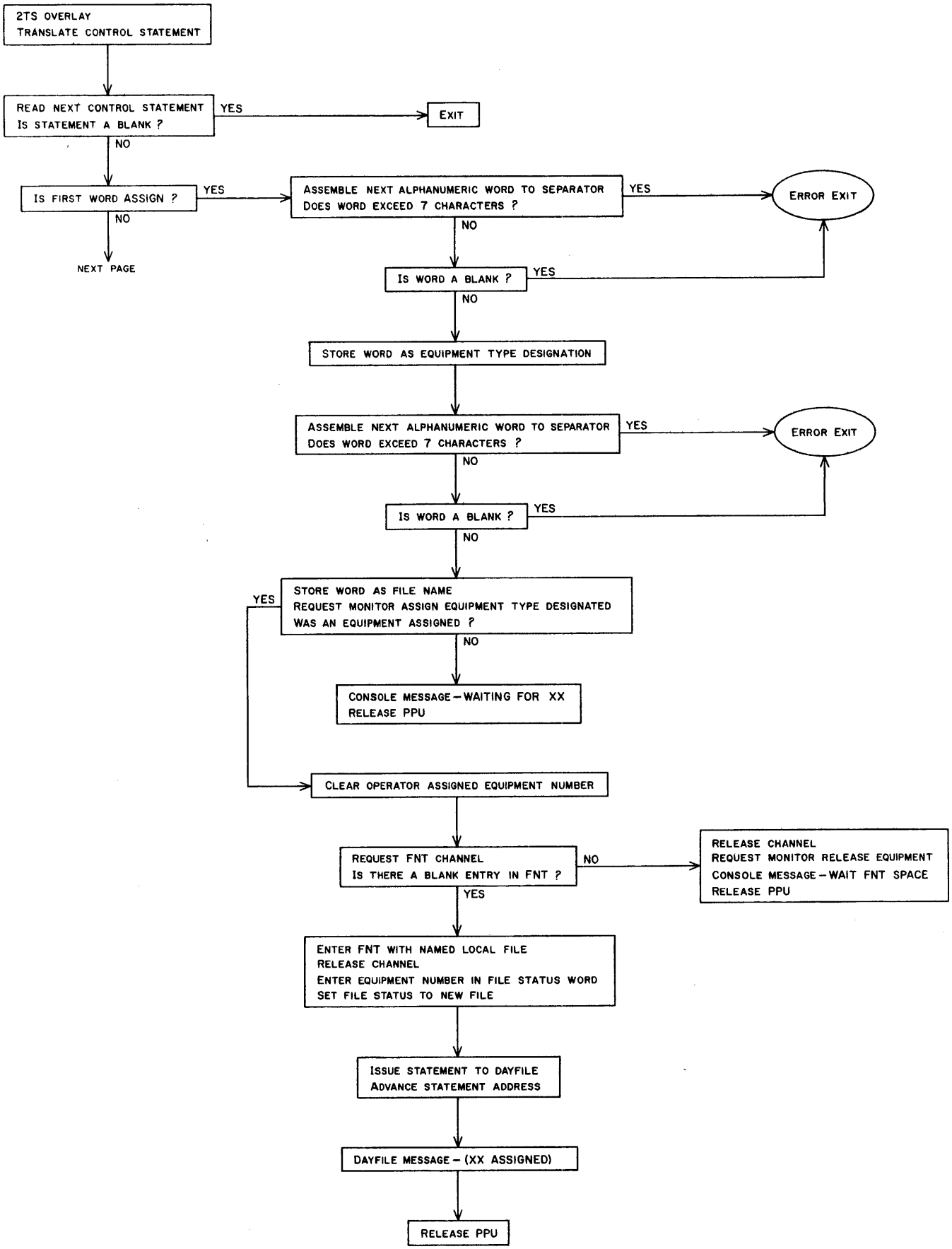

2BT SUBROUTINE
READ ONE BLOCK
REQUEST CHANNEL FOR TAPE UNIT
READ TAPE STATUS
IS TAPE READY ?
NO
CONSOLE MESSAGE - TAPE XX NOT READY
RELEASE CHANNEL
PAUSE FOR MONITOR
READ RA
IS ERROR FLAG SET ?
YES
RELEASE PPU
YES
NO
NO
READ TAPE
IS PARITY CHECK OK ?
YES
NO
REQUEST CHANNEL FOR TAPE UNIT
READ TAPE STATUS
IS TAPE READY ?
YES
RELEASE CHANNEL
ADVANCE BLOCK COUNT
EXIT
WAS A FILE MARK READ ?
YES
RELEASE CHANNEL
ADVANCE BLOCK COUNT
EXIT
NO
HAS BLOCK BEEN READ 3 TIMES ?
YES
DAYFILE MESSAGE - TAPE XX PARITY ERROR
RELEASE CHANNEL
ABORT CONTROL POINT
RELEASE PPU
NO
BACKSPACE TAPE


2DF OVERLAY
DROP FILE
FILE NAME IN 40/44
SEARCH FNT FOR NAME
IS FILE NAME IN FNT ?
NO
EXIT
YES
READ FILE STATUS
IS AN EQUIPMENT ASSIGNED ?
YES
REQUEST MONITOR DROP EQUIPMENT
NO
IS FILE IN COMMON STATUS ?
YES
RELEASE COMMON FILE
EXIT
NO
IS FILE A DISK FILE ?
NO
YES
CLEAR FILE STATUS
CLEAR FNT ENTRY
EXIT
HAS FILE BEEN USED ?
NO
YES
MODIFY OVERLAY FOR EQUIPMENT PARAMETERS
REQUEST CHANNEL FOR DISK FILE
REWIND FILE
POSITION DISK TO NEXT TRACK
REQUEST MONITOR RELEASE TRACK
READ NEXT DISK SECTOR
IS SECTOR A FILE MARK ?
YES
RELEASE CHANNEL
CLEAR FILE STATUS
CLEAR FNT ENTRY
EXIT
NO
NO
YES
IS SECTOR THE LAST SECTOR IN THIS TRACK ?

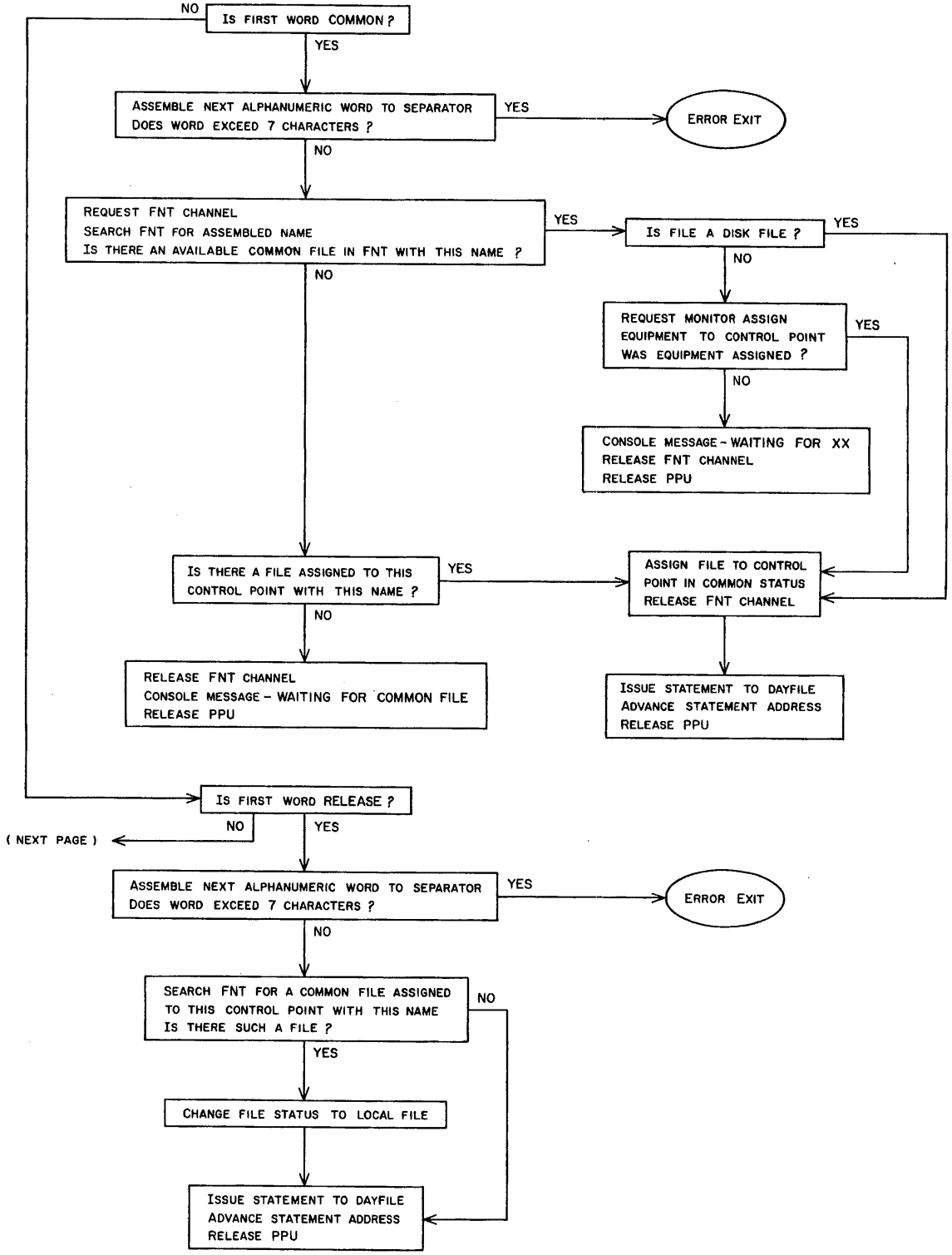

2DT OVERLAY
DROP DISK TRACKS
FILE STATUS IN 20/24
MODIFY OVERLAY FOR EQUIPMENT PARAMETERS
HAS FILE BEEN USED ?
NO
EXIT
YES
HOLD CURRENT TRACK NUMBER AND SECTOR NUMBER
REQUEST CHANNEL FOR DISK FILE
POSITION DISK FOR NEXT SECTOR
READ NEXT SECTOR
IS SECTOR A FILE MARK ?
YES
RELEASE CHANNEL
RESTORE TRACK AND SECTOR NUMBER
EXIT
NO
NO
IS SECTOR THE LAST SECTOR
IN THIS TRACK ?
YES
REQUEST MONITOR RELEASE NEXT TRACK

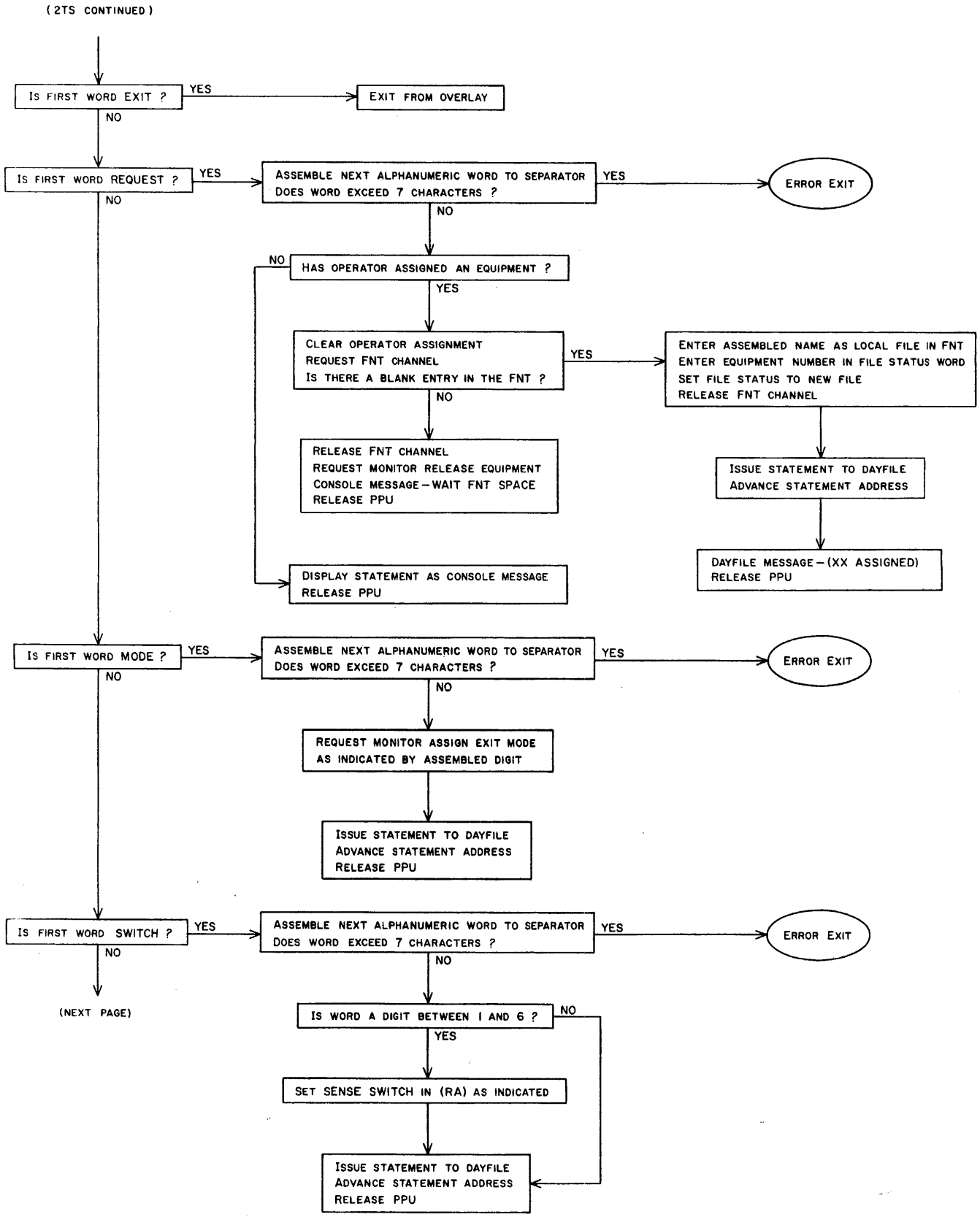

2EF OVERLAY
PROCESS ERROR FLAG
CLEAR ERROR FLAG AT THE CONTROL POINT
IS A TIME LIMIT INDICATED ?
YES
DAYFILE MESSAGE-TIME LIMIT.
NO
IS AN ARITHMETIC ERROR INDICATED ?
YES
DAYFILE MESSAGE-ARITH. ERROR.
NO
IS A PPU CALL ERROR INDICATED ?
YES
DAYFILE MESSAGE- PP CALL ERROR.
NO
IS AN OPERATOR DROP INDICATED ?
YES
DAYFILE MESSAGE-OPERATOR DROP.
NO
IS A TRACK LIMIT INDICATED ?
YES
DAYFILE MESSAGE-TRACK LIMIT.
NO
READ NEXT CONTROL STATEMENT
IS STATEMENT A BLANK ?
YES
EXIT
NO
NO
IS STATEMENT AN EXIT ?
YES
ADVANCE TO NEXT CONTROL STATEMENT
EXIT

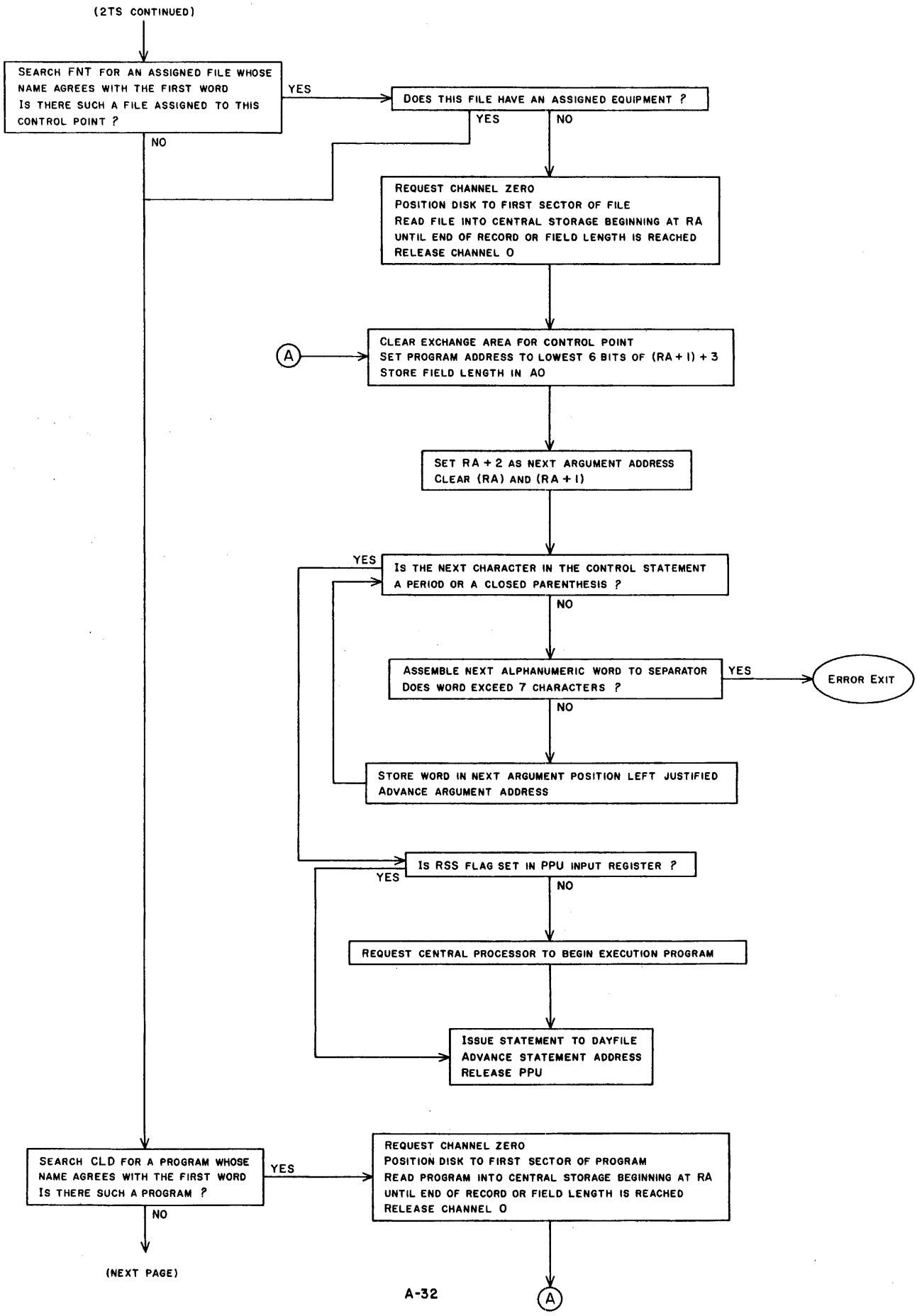

2TJ OVERLAY
TRANSLATE JOB NAME
ARE THERE MORE THAN 95 WORDS IN THE CIRCULAR BUFFER ?
YES
DAYFILE MESSAGE-TOO MANY CONTROL CARDS
RELEASE PPU
NO
READ FIRST CONTROL CARD
ASSEMBLE ALPHANUMERIC WORD TO SEPARATOR
DOES WORD EXCEED 7 CHARACTERS ?
YES
DAYFILE MESSAGE - JOB CARD ERROR
RELEASE PPU
NO
IS FIRST CHARACTER A LETTER ?
NO
YES
STORE WORD AS NAME OF JOB
SET PRIORITY 1.
SET TIME LIMIT ONE MINUTE
SET FIELD LENGTH 40000
EXIT
WAS SEPARATOR A BLANK ?
YES
NO
ASSEMBLE NEXT ALPHANUMERIC WORD TO SEPARATOR
DOES WORD EXCEED 7 CHARACTERS ?
YES
DAYFILE MESSAGE - JOB CARD ERROR
RELEASE PPU
NO
CONVERT OCTAL CHARACTERS TO BINARY
ARE LOWEST ORDER 4 BITS ZERO ?
YES
SET PRIORITY 1.
SET TIME LIMIT ONE MINUTE
SET FIELD LENGTH 40000
EXIT
NO
STORE LOWEST ORDER 4 BITS AS PRIORITY
SET TIME LIMIT ONE MINUTE
SET FIELD LENGTH 40000
EXIT
WAS SEPARATOR A BLANK ?
YES
NO
ASSEMBLE NEXT ALPHANUMERIC WORD TO SEPARATOR
DOES WORD EXCEED 7 CHARACTERS ?
YES
DAYFILE MESSAGE - JOB CARD ERROR
RELEASE PPU
NO
CONVERT OCTAL CHARACTERS TO BINARY
ROUND UP LAST OCTAL DIGIT
STORE LOWEST ORDER 5 OCTAL DIGITS AS TIME LIMIT
IS TIME LIMIT ZERO ?
YES
SET TIME LIMIT ONE MINUTE
NO
WAS SEPARATOR A BLANK ?
YES
SET FIELD LENGTH 40000
EXIT
NO
ASSEMBLE NEXT ALPHANUMERIC WORD TO SEPARATOR
DOES WORD EXCEED 7 CHARACTERS ?
YES
DAYFILE MESSAGE - JOB CARD ERROR
RELEASE PPU
NO
CONVERT OCTAL CHARACTERS TO BINARY
ROUND UP LAST TWO OCTAL DIGITS
STORE LOWEST ORDER 17 BITS AS FIELD LENGTH
IS FIELD LENGTH ZERO ?
YES
SET FIELD LENGTH 40000
EXIT
NO
CLEAR PPU TIME CHARGES TO CONTROL POINT
EXIT

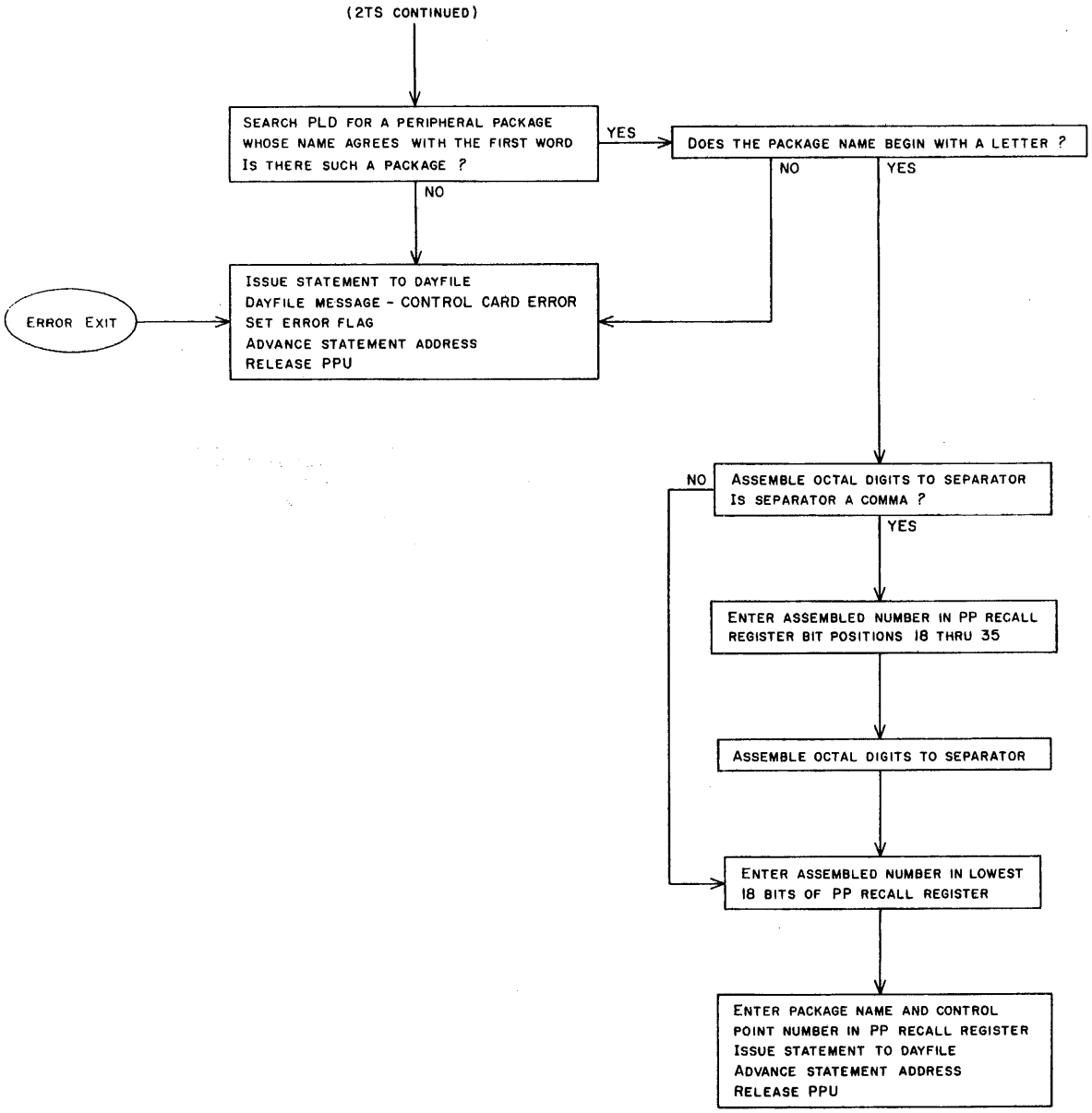

2TS OVERLAY
TRANSLATE CONTROL STATEMENT
READ NEXT CONTROL STATEMENT
IS STATEMENT A BLANK ?
YES
EXIT
NO
IS FIRST WORD ASSIGN ?
YES
NO
NEXT PAGE
ASSEMBLE NEXT ALPHANUMERIC WORD TO SEPARATOR
DOES WORD EXCEED 7 CHARACTERS ?
YES
ERROR EXIT
NO
IS WORD A BLANK ?
YES
NO
STORE WORD AS EQUIPMENT TYPE DESIGNATION
ASSEMBLE NEXT ALPHANUMERIC WORD TO SEPARATOR
DOES WORD EXCEED 7 CHARACTERS ?
YES
ERROR EXIT
NO
IS WORD A BLANK ?
YES
NO
YES
STORE WORD AS FILE NAME
REQUEST MONITOR ASSIGN EQUIPMENT TYPE DESIGNATED
WAS AN EQUIPMENT ASSIGNED ?
NO
CONSOLE MESSAGE – WAITING FOR XX
RELEASE PPU
CLEAR OPERATOR ASSIGNED EQUIPMENT NUMBER
REQUEST FNT CHANNEL
IS THERE A BLANK ENTRY IN FNT ?
NO
RELEASE CHANNEL
REQUEST MONITOR RELEASE EQUIPMENT
CONSOLE MESSAGE – WAIT FNT SPACE
RELEASE PPU
YES
ENTER FNT WITH NAMED LOCAL FILE
RELEASE CHANNEL
ENTER EQUIPMENT NUMBER IN FILE STATUS WORD
SET FILE STATUS TO NEW FILE
ISSUE STATEMENT TO DAYFILE
ADVANCE STATEMENT ADDRESS
DAYFILE MESSAGE – (XX ASSIGNED)
RELEASE PPU

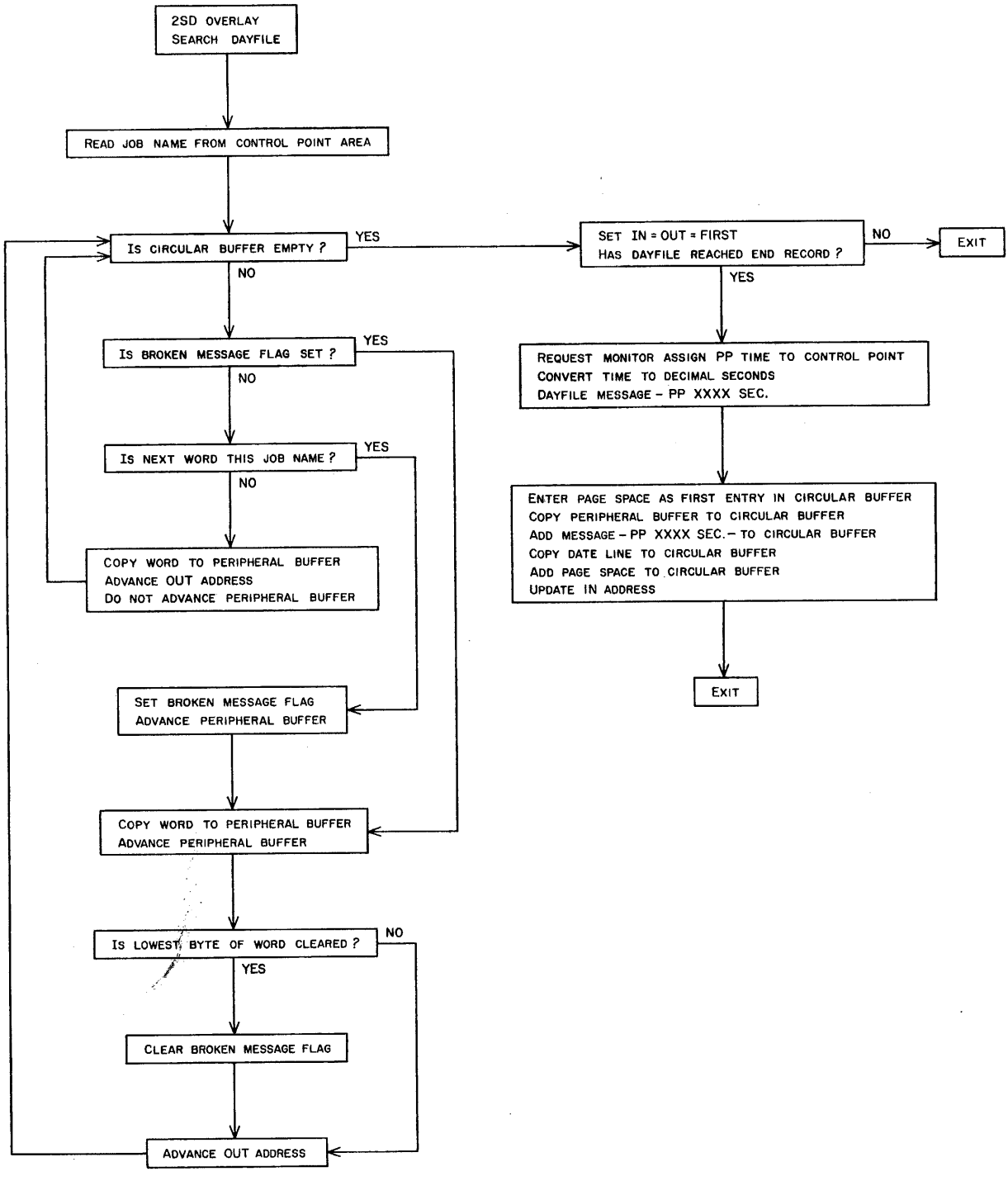
(2TS CONTINED)

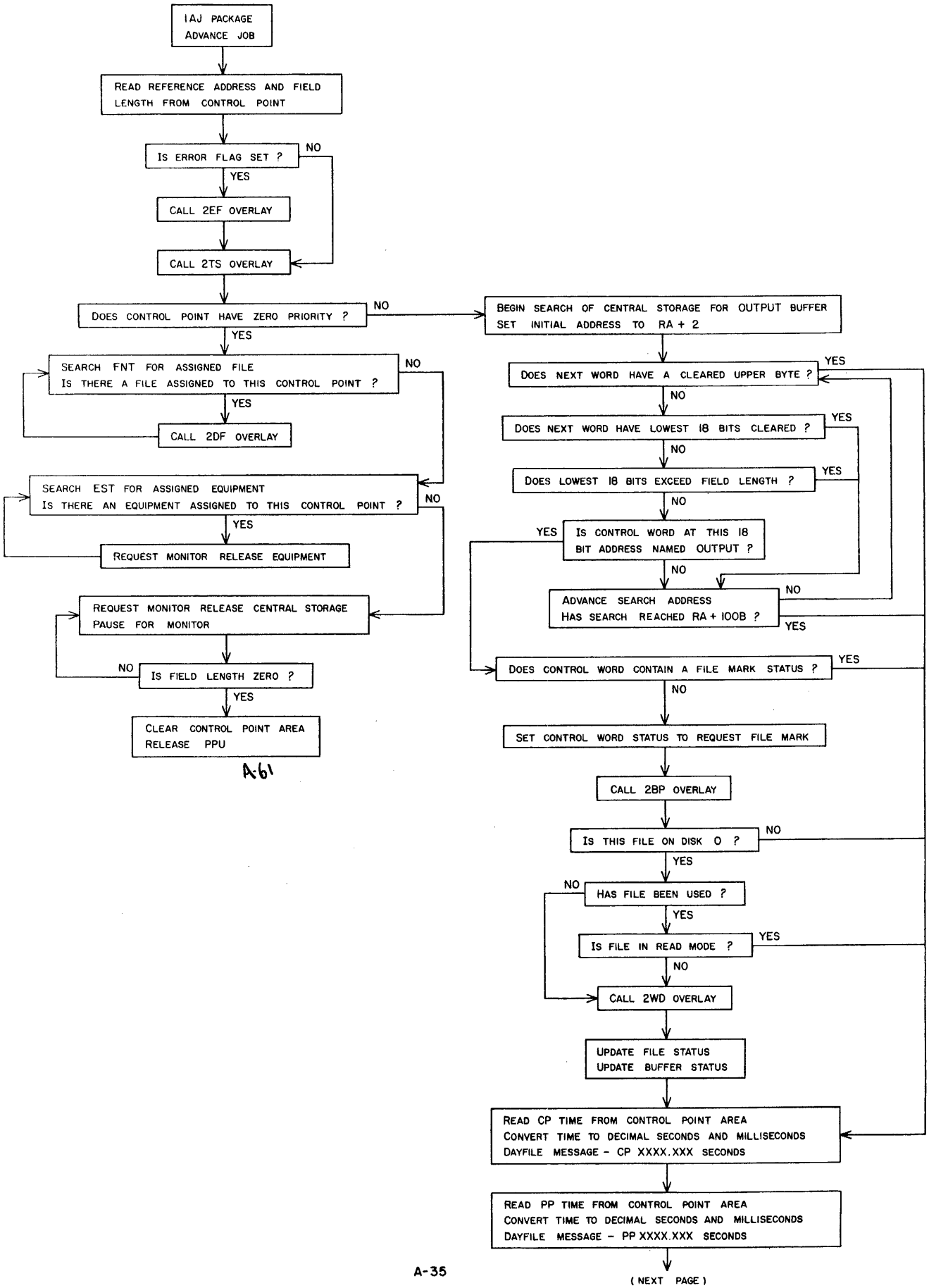


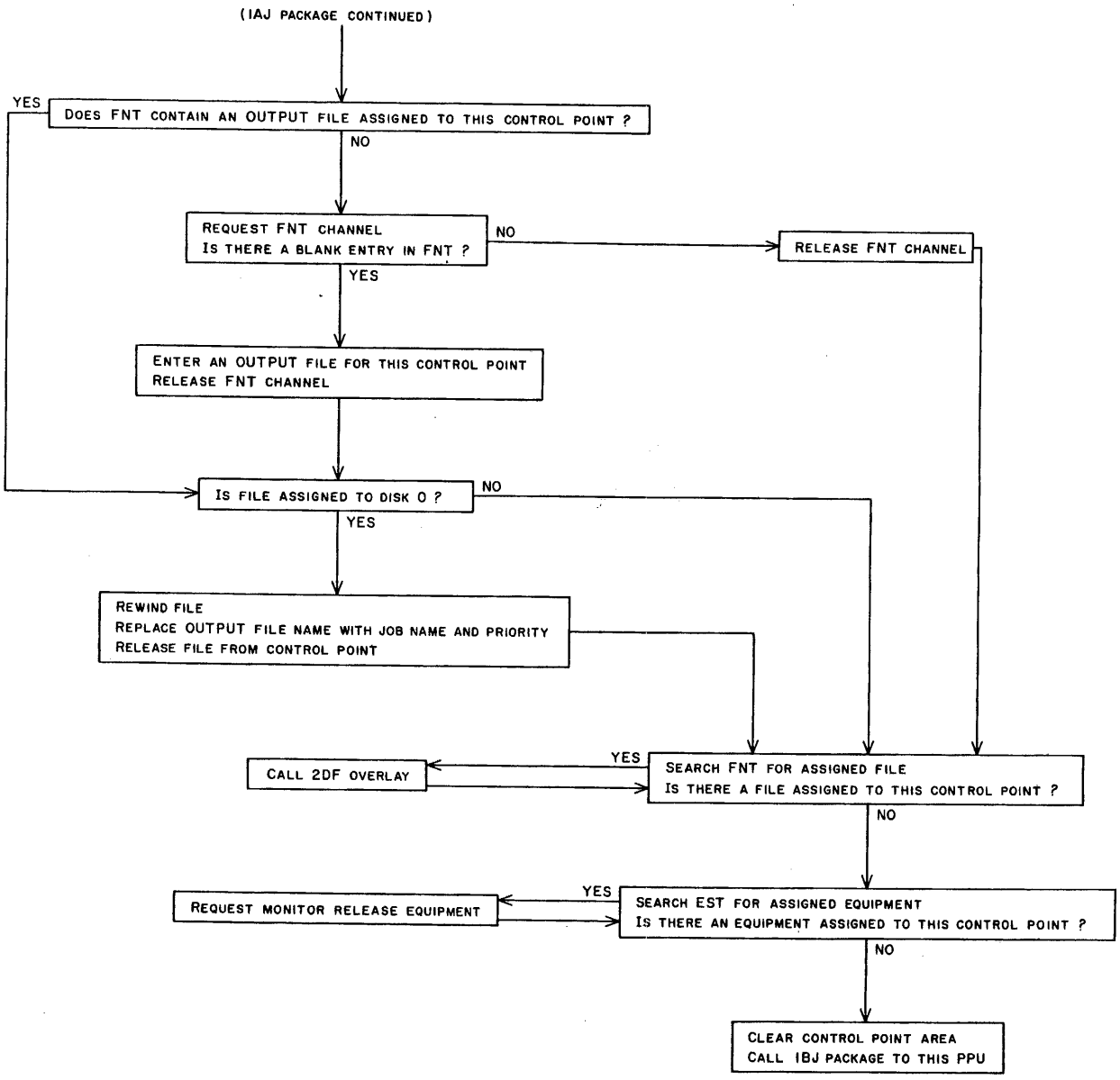

(2TS CONTINUED)
IS FIRST WORD EXIT ?
YES
EXIT FROM OVERLAY
NO
IS FIRST WORD REQUEST ?
YES
ASSEMBLE NEXT ALPHANUMERIC WORD TO SEPARATOR
DOES WORD EXCEED 7 CHARACTERS ?
YES
ERROR EXIT
NO
NO
HAS OPERATOR ASSIGNED AN EQUIPMENT ?
YES
CLEAR OPERATOR ASSIGNMENT
REQUEST FNT CHANNEL
IS THERE A BLANK ENTRY IN THE FNT ?
YES
ENTER ASSEMBLED NAME AS LOCAL FILE IN FNT
ENTER EQUIPMENT NUMBER IN FILE STATUS WORD
SET FILE STATUS TO NEW FILE
RELEASE FNT CHANNEL
NO
RELEASE FNT CHANNEL
REQUEST MONITOR RELEASE EQUIPMENT
CONSOLE MESSAGE - WAIT FNT SPACE
RELEASE PPU
ISSUE STATEMENT TO DAYFILE
ADVANCE STATEMENT ADDRESS
DAYFILE MESSAGE - (XX ASSIGNED)
RELEASE PPU
DISPLAY STATEMENT AS CONSOLE MESSAGE
RELEASE PPU
IS FIRST WORD MODE ?
YES
ASSEMBLE NEXT ALPHANUMERIC WORD TO SEPARATOR
DOES WORD EXCEED 7 CHARACTERS ?
YES
ERROR EXIT
NO
NO
REQUEST MONITOR ASSIGN EXIT MODE
AS INDICATED BY ASSEMBLED DIGIT
ISSUE STATEMENT TO DAYFILE
ADVANCE STATEMENT ADDRESS
RELEASE PPU
IS FIRST WORD SWITCH ?
YES
ASSEMBLE NEXT ALPHANUMERIC WORD TO SEPARATOR
DOES WORD EXCEED 7 CHARACTERS ?
YES
ERROR EXIT
NO
NO
(NEXT PAGE)
IS WORD A DIGIT BETWEEN 1 AND 6 ?
NO
YES
SET SENSE SWITCH IN (RA) AS INDICATED
ISSUE STATEMENT TO DAYFILE
ADVANCE STATEMENT ADDRESS
RELEASE PPU

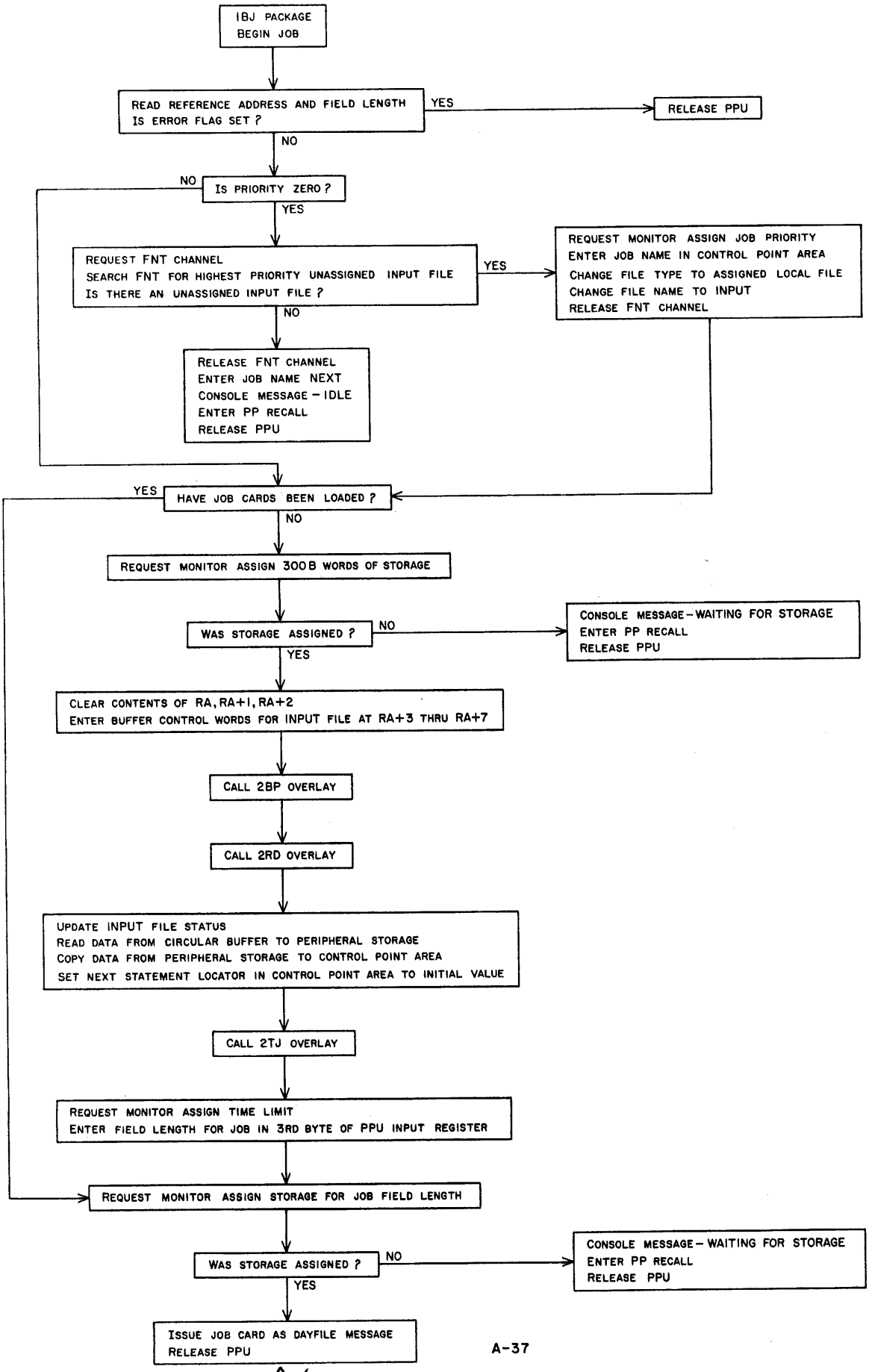

(2TS CONTINUED)
SEARCH FNT FOR AN ASSIGNED FILE WHOSE NAME AGREES WITH THE FIRST WORD
IS THERE SUCH A FILE ASSIGNED TO THIS CONTROL POINT ?
YES
NO
DOES THIS FILE HAVE AN ASSIGNED EQUIPMENT ?
YES
NO
REQUEST CHANNEL ZERO
POSITION DISK TO FIRST SECTOR OF FILE
READ FILE INTO CENTRAL STORAGE BEGINNING AT RA
UNTIL END OF RECORD OR FIELD LENGTH IS REACHED
RELEASE CHANNEL 0
A
CLEAR EXCHANGE AREA FOR CONTROL POINT
SET PROGRAM ADDRESS TO LOWEST 6 BITS OF (RA + 1) + 3
STORE FIELD LENGTH IN A0
SET RA + 2 AS NEXT ARGUMENT ADDRESS
CLEAR (RA) AND (RA + 1)
YES
IS THE NEXT CHARACTER IN THE CONTROL STATEMENT A PERIOD OR A CLOSED PARENTHESIS ?
NO
ASSEMBLE NEXT ALPHANUMERIC WORD TO SEPARATOR
DOES WORD EXCEED 7 CHARACTERS ?
YES
ERROR EXIT
NO
STORE WORD IN NEXT ARGUMENT POSITION LEFT JUSTIFIED
ADVANCE ARGUMENT ADDRESS
IS RSS FLAG SET IN PPU INPUT REGISTER ?
YES
NO
REQUEST CENTRAL PROCESSOR TO BEGIN EXECUTION PROGRAM
ISSUE STATEMENT TO DAYFILE
ADVANCE STATEMENT ADDRESS
RELEASE PPU
SEARCH CLD FOR A PROGRAM WHOSE NAME AGREES WITH THE FIRST WORD
IS THERE SUCH A PROGRAM ?
YES
NO
REQUEST CHANNEL ZERO
POSITION DISK TO FIRST SECTOR OF PROGRAM
READ PROGRAM INTO CENTRAL STORAGE BEGINNING AT RA
UNTIL END OF RECORD OR FIELD LENGTH IS REACHED
RELEASE CHANNEL 0
(NEXT PAGE)
A


(2TS CONTINUED)
SEARCH PLD FOR A PERIPHERAL PACKAGE WHOSE NAME AGREES WITH THE FIRST WORD
IS THERE SUCH A PACKAGE ?
YES
DOES THE PACKAGE NAME BEGIN WITH A LETTER ?
NO
YES
NO
ISSUE STATEMENT TO DAYFILE
DAYFILE MESSAGE - CONTROL CARD ERROR
SET ERROR FLAG
ADVANCE STATEMENT ADDRESS
RELEASE PPU
ERROR EXIT
NO
ASSEMBLE OCTAL DIGITS TO SEPARATOR
IS SEPARATOR A COMMA ?
YES
ENTER ASSEMBLED NUMBER IN PP RECALL REGISTER BIT POSITIONS 18 THRU 35
ASSEMBLE OCTAL DIGITS TO SEPARATOR
ENTER ASSEMBLED NUMBER IN LOWEST 18 BITS OF PP RECALL REGISTER
ENTER PACKAGE NAME AND CONTROL POINT NUMBER IN PP RECALL REGISTER
ISSUE STATEMENT TO DAYFILE
ADVANCE STATEMENT ADDRESS
RELEASE PPU

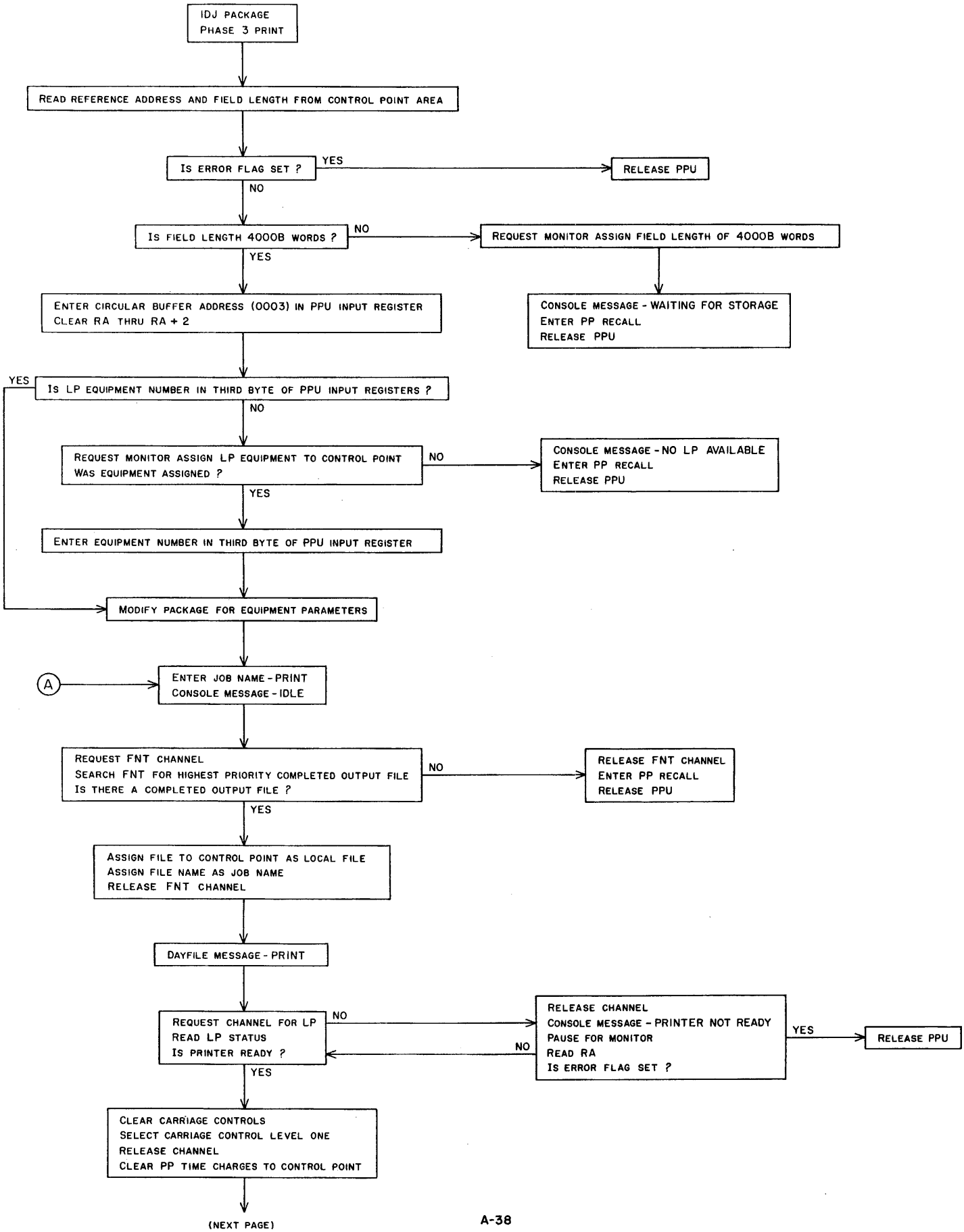

2SD OVERLAY
SEARCH DAYFILE
READ JOB NAME FROM CONTROL POINT AREA
IS CIRCULAR BUFFER EMPTY ?
YES
NO
IS BROKEN MESSAGE FLAG SET ?
YES
NO
IS NEXT WORD THIS JOB NAME ?
YES
NO
COPY WORD TO PERIPHERAL BUFFER
ADVANCE OUT ADDRESS
DO NOT ADVANCE PERIPHERAL BUFFER
SET BROKEN MESSAGE FLAG
ADVANCE PERIPHERAL BUFFER
COPY WORD TO PERIPHERAL BUFFER
ADVANCE PERIPHERAL BUFFER
IS LOWEST BYTE OF WORD CLEARED ?
NO
YES
CLEAR BROKEN MESSAGE FLAG
ADVANCE OUT ADDRESS
SET IN = OUT = FIRST
HAS DAYFILE REACHED END RECORD ?
NO
EXIT
YES
REQUEST MONITOR ASSIGN PP TIME TO CONTROL POINT
CONVERT TIME TO DECIMAL SECONDS
DAYFILE MESSAGE - PP XXXX SEC.
ENTER PAGE SPACE AS FIRST ENTRY IN CIRCULAR BUFFER
COPY PERIPHERAL BUFFER TO CIRCULAR BUFFER
ADD MESSAGE - PP XXXX SEC. - TO CIRCULAR BUFFER
COPY DATE LINE TO CIRCULAR BUFFER
ADD PAGE SPACE TO CIRCULAR BUFFER
UPDATE IN ADDRESS
EXIT

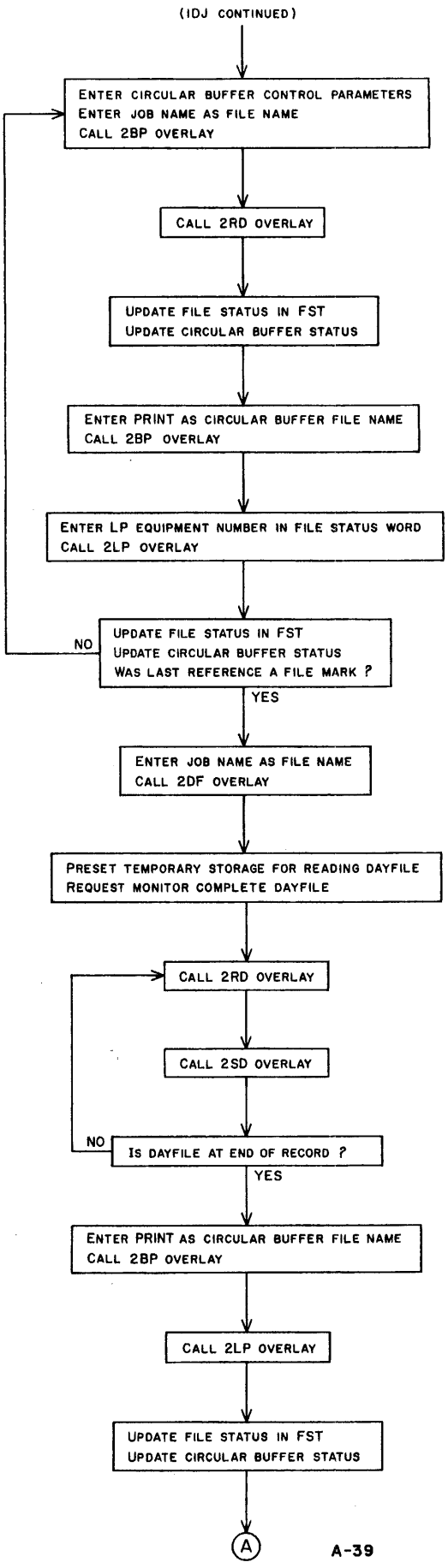

IAJ PACKAGE
ADVANCE JOB
READ REFERENCE ADDRESS AND FIELD LENGTH FROM CONTROL POINT
IS ERROR FLAG SET ?
NO
YES
CALL 2EF OVERLAY
CALL 2TS OVERLAY
DOES CONTROL POINT HAVE ZERO PRIORITY ?
NO
YES
SEARCH FNT FOR ASSIGNED FILE
IS THERE A FILE ASSIGNED TO THIS CONTROL POINT ?
NO
YES
CALL 2DF OVERLAY
SEARCH EST FOR ASSIGNED EQUIPMENT
IS THERE AN EQUIPMENT ASSIGNED TO THIS CONTROL POINT ?
NO
YES
REQUEST MONITOR RELEASE EQUIPMENT
REQUEST MONITOR RELEASE CENTRAL STORAGE
PAUSE FOR MONITOR
NO
IS FIELD LENGTH ZERO ?
YES
CLEAR CONTROL POINT AREA
RELEASE PPU
A-61
BEGIN SEARCH OF CENTRAL STORAGE FOR OUTPUT BUFFER
SET INITIAL ADDRESS TO RA + 2
DOES NEXT WORD HAVE A CLEARED UPPER BYTE ?
YES
NO
DOES NEXT WORD HAVE LOWEST 18 BITS CLEARED ?
YES
NO
DOES LOWEST 18 BITS EXCEED FIELD LENGTH ?
YES
NO
YES
IS CONTROL WORD AT THIS 18 BIT ADDRESS NAMED OUTPUT ?
NO
ADVANCE SEARCH ADDRESS
HAS SEARCH REACHED RA + 100B ?
NO
YES
DOES CONTROL WORD CONTAIN A FILE MARK STATUS ?
YES
NO
SET CONTROL WORD STATUS TO REQUEST FILE MARK
CALL 2BP OVERLAY
IS THIS FILE ON DISK 0 ?
NO
YES
NO
HAS FILE BEEN USED ?
YES
IS FILE IN READ MODE ?
YES
NO
CALL 2WD OVERLAY
UPDATE FILE STATUS
UPDATE BUFFER STATUS
READ CP TIME FROM CONTROL POINT AREA
CONVERT TIME TO DECIMAL SECONDS AND MILLISECONDS
DAYFILE MESSAGE - CP XXXX.XXX SECONDS
READ PP TIME FROM CONTROL POINT AREA
CONVERT TIME TO DECIMAL SECONDS AND MILLISECONDS
DAYFILE MESSAGE - PP XXXX.XXX SECONDS
(NEXT PAGE)

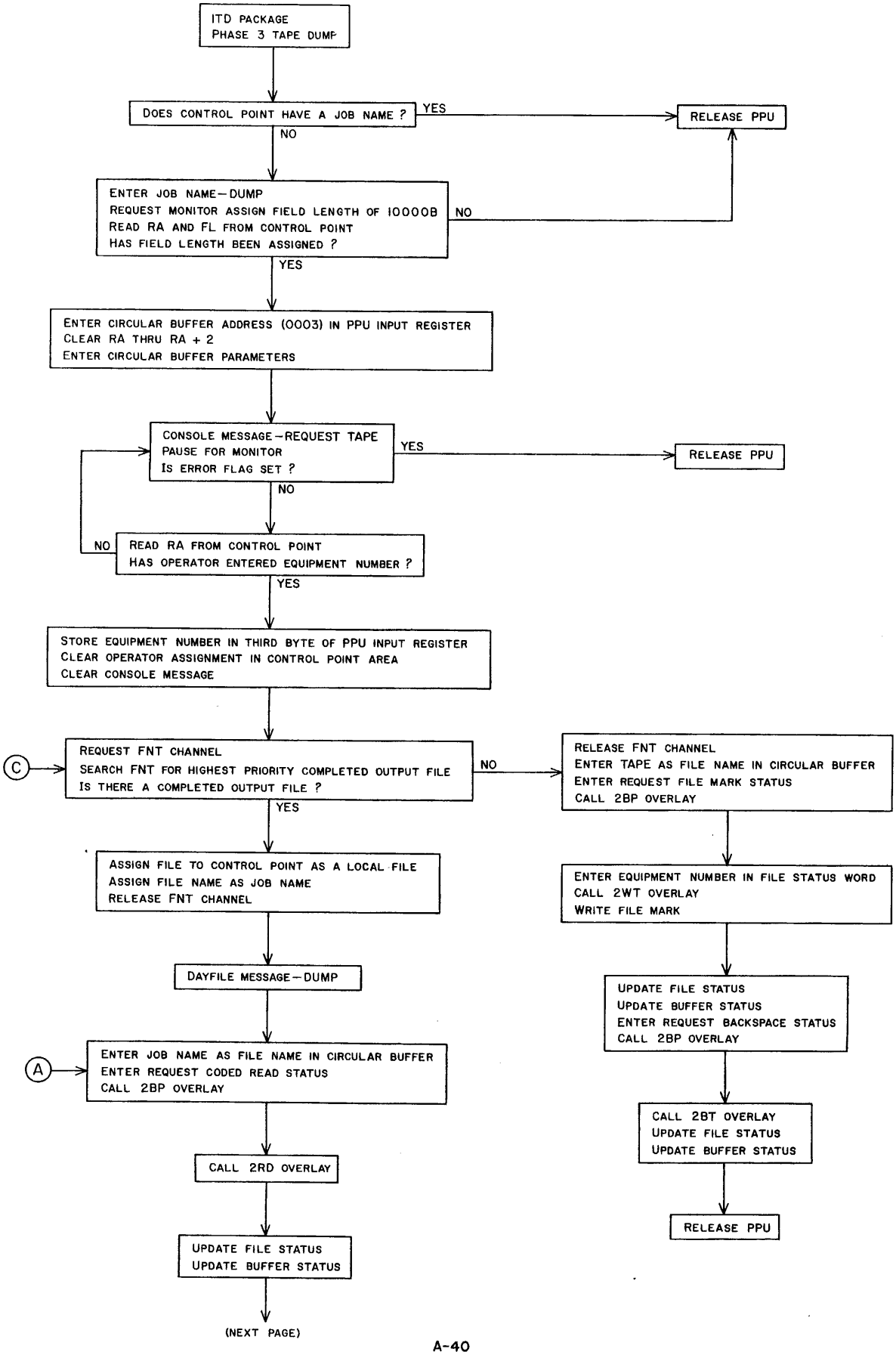

(IAJ PACKAGE CONTINUED)
YES
DOES FNT CONTAIN AN OUTPUT FILE ASSIGNED TO THIS CONTROL POINT ?
NO
REQUEST FNT CHANNEL
IS THERE A BLANK ENTRY IN FNT ?
NO
RELEASE FNT CHANNEL
YES
ENTER AN OUTPUT FILE FOR THIS CONTROL POINT
RELEASE FNT CHANNEL
IS FILE ASSIGNED TO DISK 0 ?
NO
YES
REWIND FILE
REPLACE OUTPUT FILE NAME WITH JOB NAME AND PRIORITY
RELEASE FILE FROM CONTROL POINT
YES
CALL 2DF OVERLAY
SEARCH FNT FOR ASSIGNED FILE
IS THERE A FILE ASSIGNED TO THIS CONTROL POINT ?
NO
YES
REQUEST MONITOR RELEASE EQUIPMENT
SEARCH EST FOR ASSIGNED EQUIPMENT
IS THERE AN EQUIPMENT ASSIGNED TO THIS CONTROL POINT ?
NO
CLEAR CONTROL POINT AREA
CALL IBJ PACKAGE TO THIS PPU

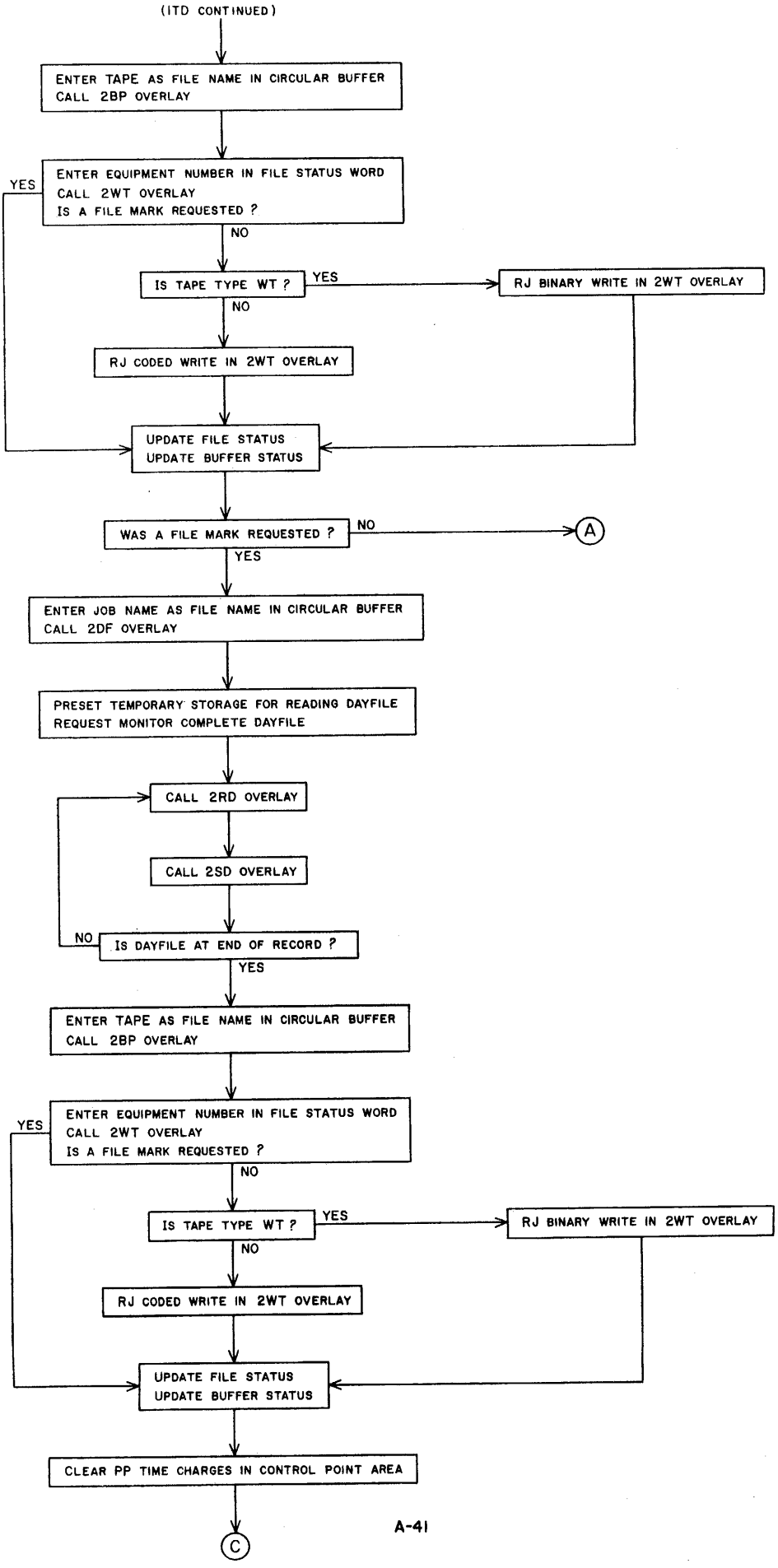

IBJ PACKAGE
BEGIN JOB
READ REFERENCE ADDRESS AND FIELD LENGTH
IS ERROR FLAG SET ?
YES
RELEASE PPU
NO
NO
IS PRIORITY ZERO ?
YES
REQUEST FNT CHANNEL
SEARCH FNT FOR HIGHEST PRIORITY UNASSIGNED INPUT FILE
IS THERE AN UNASSIGNED INPUT FILE ?
YES
REQUEST MONITOR ASSIGN JOB PRIORITY
ENTER JOB NAME IN CONTROL POINT AREA
CHANGE FILE TYPE TO ASSIGNED LOCAL FILE
CHANGE FILE NAME TO INPUT
RELEASE FNT CHANNEL
NO
RELEASE FNT CHANNEL
ENTER JOB NAME NEXT
CONSOLE MESSAGE - IDLE
ENTER PP RECALL
RELEASE PPU
YES
HAVE JOB CARDS BEEN LOADED ?
NO
REQUEST MONITOR ASSIGN 300B WORDS OF STORAGE
WAS STORAGE ASSIGNED ?
NO
CONSOLE MESSAGE-WAITING FOR STORAGE
ENTER PP RECALL
RELEASE PPU
YES
CLEAR CONTENTS OF RA, RA+1, RA+2
ENTER BUFFER CONTROL WORDS FOR INPUT FILE AT RA+3 THRU RA+7
CALL 2BP OVERLAY
CALL 2RD OVERLAY
UPDATE INPUT FILE STATUS
READ DATA FROM CIRCULAR BUFFER TO PERIPHERAL STORAGE
COPY DATA FROM PERIPHERAL STORAGE TO CONTROL POINT AREA
SET NEXT STATEMENT LOCATOR IN CONTROL POINT AREA TO INITIAL VALUE
CALL 2TJ OVERLAY
REQUEST MONITOR ASSIGN TIME LIMIT
ENTER FIELD LENGTH FOR JOB IN 3RD BYTE OF PPU INPUT REGISTER
REQUEST MONITOR ASSIGN STORAGE FOR JOB FIELD LENGTH
WAS STORAGE ASSIGNED ?
NO
CONSOLE MESSAGE-WAITING FOR STORAGE
ENTER PP RECALL
RELEASE PPU
YES
ISSUE JOB CARD AS DAYFILE MESSAGE
RELEASE PPU

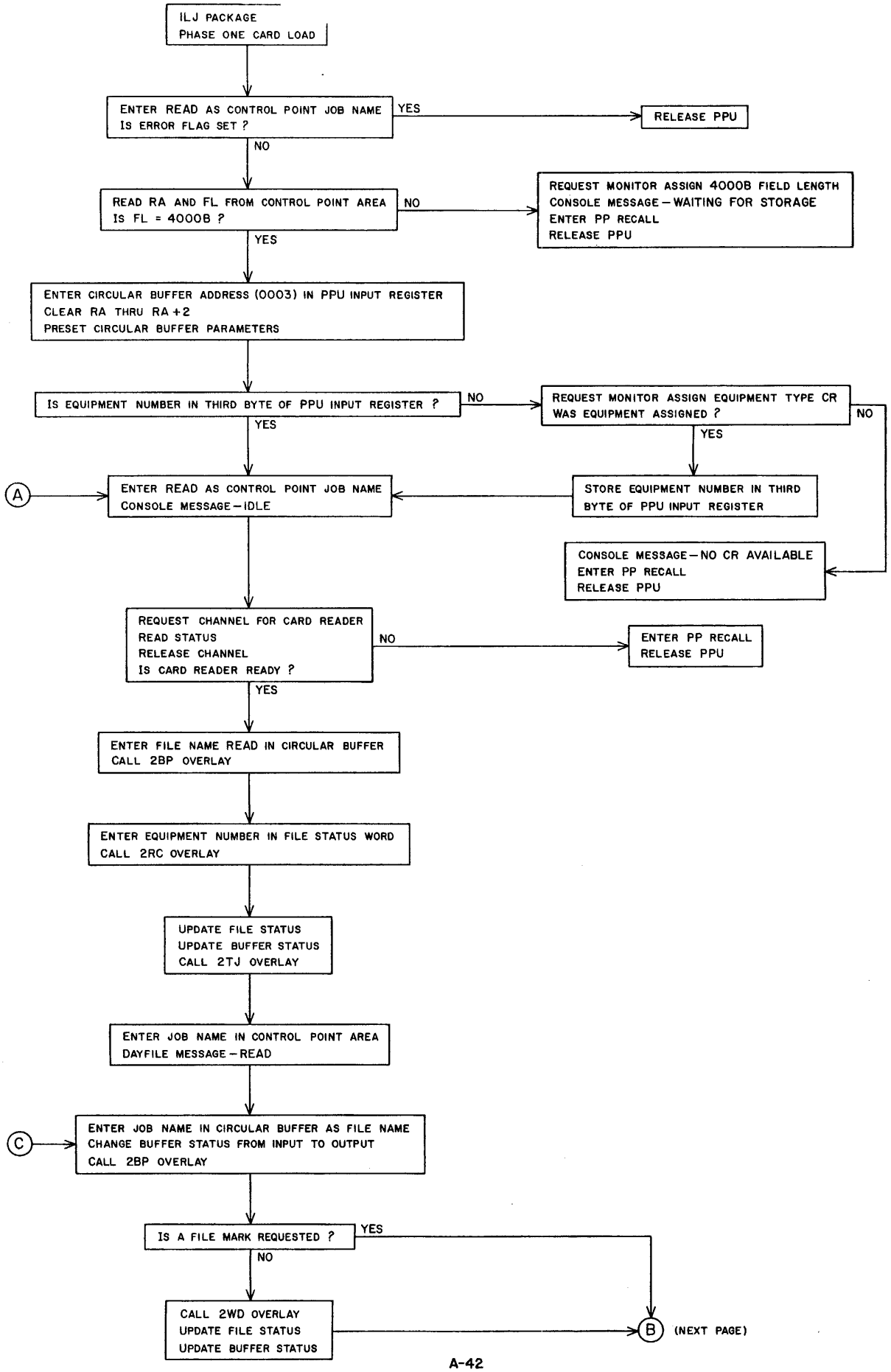
A-61


IDJ PACKAGE
PHASE 3 PRINT
READ REFERENCE ADDRESS AND FIELD LENGTH FROM CONTROL POINT AREA
IS ERROR FLAG SET ?
YES
RELEASE PPU
NO
IS FIELD LENGTH 4000B WORDS ?
NO
REQUEST MONITOR ASSIGN FIELD LENGTH OF 4000B WORDS
YES
CONSOLE MESSAGE - WAITING FOR STORAGE
ENTER PP RECALL
RELEASE PPU
ENTER CIRCULAR BUFFER ADDRESS (0003) IN PPU INPUT REGISTER
CLEAR RA THRU RA + 2
YES
IS LP EQUIPMENT NUMBER IN THIRD BYTE OF PPU INPUT REGISTERS ?
NO
REQUEST MONITOR ASSIGN LP EQUIPMENT TO CONTROL POINT
WAS EQUIPMENT ASSIGNED ?
NO
CONSOLE MESSAGE - NO LP AVAILABLE
ENTER PP RECALL
RELEASE PPU
YES
ENTER EQUIPMENT NUMBER IN THIRD BYTE OF PPU INPUT REGISTER
MODIFY PACKAGE FOR EQUIPMENT PARAMETERS
A
ENTER JOB NAME - PRINT
CONSOLE MESSAGE - IDLE
REQUEST FNT CHANNEL
SEARCH FNT FOR HIGHEST PRIORITY COMPLETED OUTPUT FILE
IS THERE A COMPLETED OUTPUT FILE ?
NO
RELEASE FNT CHANNEL
ENTER PP RECALL
RELEASE PPU
YES
ASSIGN FILE TO CONTROL POINT AS LOCAL FILE
ASSIGN FILE NAME AS JOB NAME
RELEASE FNT CHANNEL
DAYFILE MESSAGE - PRINT
REQUEST CHANNEL FOR LP
READ LP STATUS
IS PRINTER READY ?
NO
RELEASE CHANNEL
CONSOLE MESSAGE - PRINTER NOT READY
PAUSE FOR MONITOR
READ RA
IS ERROR FLAG SET ?
YES
RELEASE PPU
NO
YES
CLEAR CARRIAGE CONTROLS
SELECT CARRIAGE CONTROL LEVEL ONE
RELEASE CHANNEL
CLEAR PP TIME CHARGES TO CONTROL POINT
(NEXT PAGE)

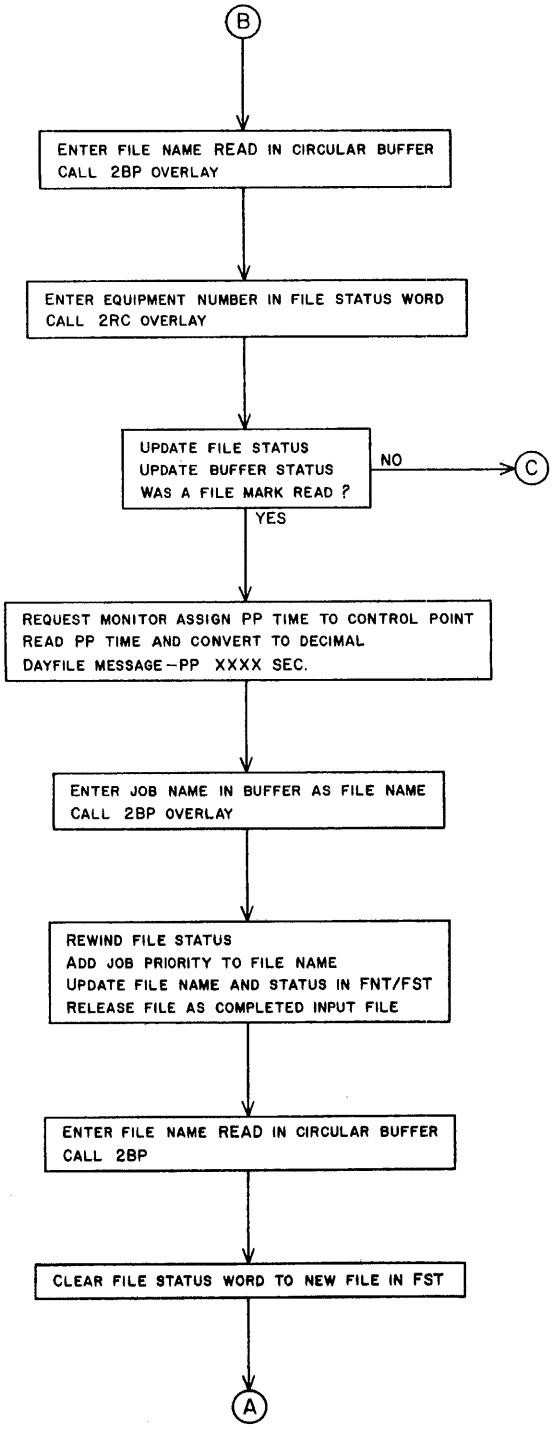

(IDJ CONTINUED)
ENTER CIRCULAR BUFFER CONTROL PARAMETERS
ENTER JOB NAME AS FILE NAME
CALL 2BP OVERLAY
CALL 2RD OVERLAY
UPDATE FILE STATUS IN FST
UPDATE CIRCULAR BUFFER STATUS
ENTER PRINT AS CIRCULAR BUFFER FILE NAME
CALL 2BP OVERLAY
ENTER LP EQUIPMENT NUMBER IN FILE STATUS WORD
CALL 2LP OVERLAY
NO
UPDATE FILE STATUS IN FST
UPDATE CIRCULAR BUFFER STATUS
WAS LAST REFERENCE A FILE MARK ?
YES
ENTER JOB NAME AS FILE NAME
CALL 2DF OVERLAY
PRESET TEMPORARY STORAGE FOR READING DAYFILE
REQUEST MONITOR COMPLETE DAYFILE
CALL 2RD OVERLAY
CALL 2SD OVERLAY
NO
IS DAYFILE AT END OF RECORD ?
YES
ENTER PRINT AS CIRCULAR BUFFER FILE NAME
CALL 2BP OVERLAY
CALL 2LP OVERLAY
UPDATE FILE STATUS IN FST
UPDATE CIRCULAR BUFFER STATUS
A

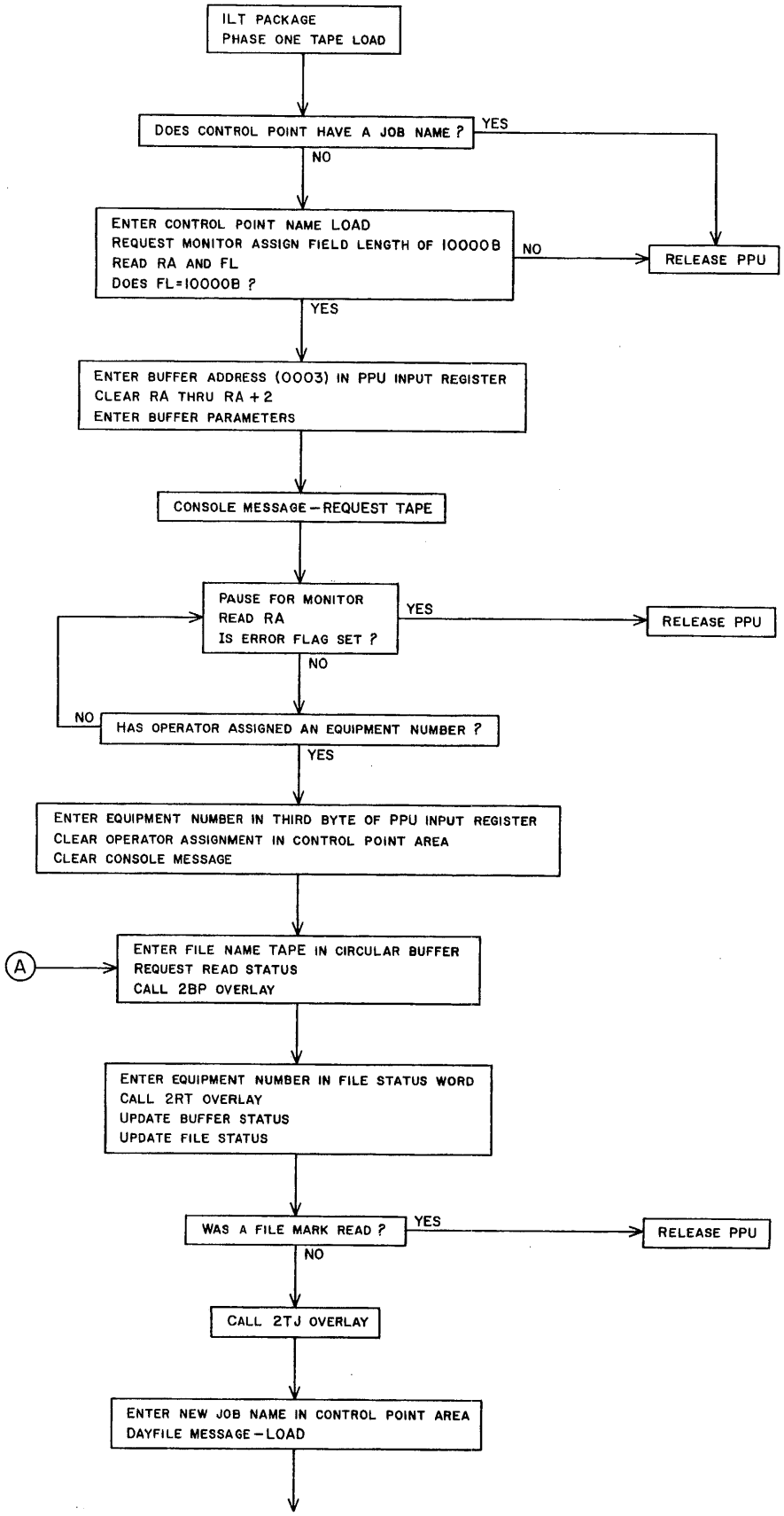

ITD PACKAGE
PHASE 3 TAPE DUMP
DOES CONTROL POINT HAVE A JOB NAME ?
YES
RELEASE PPU
NO
ENTER JOB NAME—DUMP
REQUEST MONITOR ASSIGN FIELD LENGTH OF 10000B
READ RA AND FL FROM CONTROL POINT
HAS FIELD LENGTH BEEN ASSIGNED ?
NO
YES
ENTER CIRCULAR BUFFER ADDRESS (0003) IN PPU INPUT REGISTER
CLEAR RA THRU RA + 2
ENTER CIRCULAR BUFFER PARAMETERS
CONSOLE MESSAGE—REQUEST TAPE
PAUSE FOR MONITOR
IS ERROR FLAG SET ?
YES
RELEASE PPU
NO
NO
READ RA FROM CONTROL POINT
HAS OPERATOR ENTERED EQUIPMENT NUMBER ?
YES
STORE EQUIPMENT NUMBER IN THIRD BYTE OF PPU INPUT REGISTER
CLEAR OPERATOR ASSIGNMENT IN CONTROL POINT AREA
CLEAR CONSOLE MESSAGE
C
REQUEST FNT CHANNEL
SEARCH FNT FOR HIGHEST PRIORITY COMPLETED OUTPUT FILE
IS THERE A COMPLETED OUTPUT FILE ?
NO
RELEASE FNT CHANNEL
ENTER TAPE AS FILE NAME IN CIRCULAR BUFFER
ENTER REQUEST FILE MARK STATUS
CALL 2BP OVERLAY
YES
ASSIGN FILE TO CONTROL POINT AS A LOCAL FILE
ASSIGN FILE NAME AS JOB NAME
RELEASE FNT CHANNEL
ENTER EQUIPMENT NUMBER IN FILE STATUS WORD
CALL 2WT OVERLAY
WRITE FILE MARK
DAYFILE MESSAGE—DUMP
UPDATE FILE STATUS
UPDATE BUFFER STATUS
ENTER REQUEST BACKSPACE STATUS
CALL 2BP OVERLAY
A
ENTER JOB NAME AS FILE NAME IN CIRCULAR BUFFER
ENTER REQUEST CODED READ STATUS
CALL 2BP OVERLAY
CALL 2BT OVERLAY
UPDATE FILE STATUS
UPDATE BUFFER STATUS
CALL 2RD OVERLAY
RELEASE PPU
UPDATE FILE STATUS
UPDATE BUFFER STATUS
(NEXT PAGE)


(ITD CONTINUED)
ENTER TAPE AS FILE NAME IN CIRCULAR BUFFER
CALL 2BP OVERLAY
ENTER EQUIPMENT NUMBER IN FILE STATUS WORD
CALL 2WT OVERLAY
IS A FILE MARK REQUESTED ?
YES
NO
IS TAPE TYPE WT ?
YES
RJ BINARY WRITE IN 2WT OVERLAY
NO
RJ CODED WRITE IN 2WT OVERLAY
UPDATE FILE STATUS
UPDATE BUFFER STATUS
WAS A FILE MARK REQUESTED ?
NO
A
YES
ENTER JOB NAME AS FILE NAME IN CIRCULAR BUFFER
CALL 2DF OVERLAY
PRESET TEMPORARY STORAGE FOR READING DAYFILE
REQUEST MONITOR COMPLETE DAYFILE
CALL 2RD OVERLAY
CALL 2SD OVERLAY
NO
IS DAYFILE AT END OF RECORD ?
YES
ENTER TAPE AS FILE NAME IN CIRCULAR BUFFER
CALL 2BP OVERLAY
ENTER EQUIPMENT NUMBER IN FILE STATUS WORD
CALL 2WT OVERLAY
IS A FILE MARK REQUESTED ?
YES
NO
IS TAPE TYPE WT ?
YES
RJ BINARY WRITE IN 2WT OVERLAY
NO
RJ CODED WRITE IN 2WT OVERLAY
UPDATE FILE STATUS
UPDATE BUFFER STATUS
CLEAR PP TIME CHARGES IN CONTROL POINT AREA
C


ILJ PACKAGE
PHASE ONE CARD LOAD
ENTER READ AS CONTROL POINT JOB NAME
IS ERROR FLAG SET ?
YES
RELEASE PPU
NO
READ RA AND FL FROM CONTROL POINT AREA
IS FL = 4000B ?
NO
REQUEST MONITOR ASSIGN 4000B FIELD LENGTH
CONSOLE MESSAGE – WAITING FOR STORAGE
ENTER PP RECALL
RELEASE PPU
YES
ENTER CIRCULAR BUFFER ADDRESS (0003) IN PPU INPUT REGISTER
CLEAR RA THRU RA+2
PRESET CIRCULAR BUFFER PARAMETERS
IS EQUIPMENT NUMBER IN THIRD BYTE OF PPU INPUT REGISTER ?
NO
REQUEST MONITOR ASSIGN EQUIPMENT TYPE CR
WAS EQUIPMENT ASSIGNED ?
NO
YES
YES
A
ENTER READ AS CONTROL POINT JOB NAME
CONSOLE MESSAGE – IDLE
STORE EQUIPMENT NUMBER IN THIRD
BYTE OF PPU INPUT REGISTER
CONSOLE MESSAGE – NO CR AVAILABLE
ENTER PP RECALL
RELEASE PPU
REQUEST CHANNEL FOR CARD READER
READ STATUS
RELEASE CHANNEL
IS CARD READER READY ?
NO
ENTER PP RECALL
RELEASE PPU
YES
ENTER FILE NAME READ IN CIRCULAR BUFFER
CALL 2BP OVERLAY
ENTER EQUIPMENT NUMBER IN FILE STATUS WORD
CALL 2RC OVERLAY
UPDATE FILE STATUS
UPDATE BUFFER STATUS
CALL 2TJ OVERLAY
ENTER JOB NAME IN CONTROL POINT AREA
DAYFILE MESSAGE – READ
C
ENTER JOB NAME IN CIRCULAR BUFFER AS FILE NAME
CHANGE BUFFER STATUS FROM INPUT TO OUTPUT
CALL 2BP OVERLAY
IS A FILE MARK REQUESTED ?
YES
NO
CALL 2WD OVERLAY
UPDATE FILE STATUS
UPDATE BUFFER STATUS
B
(NEXT PAGE)

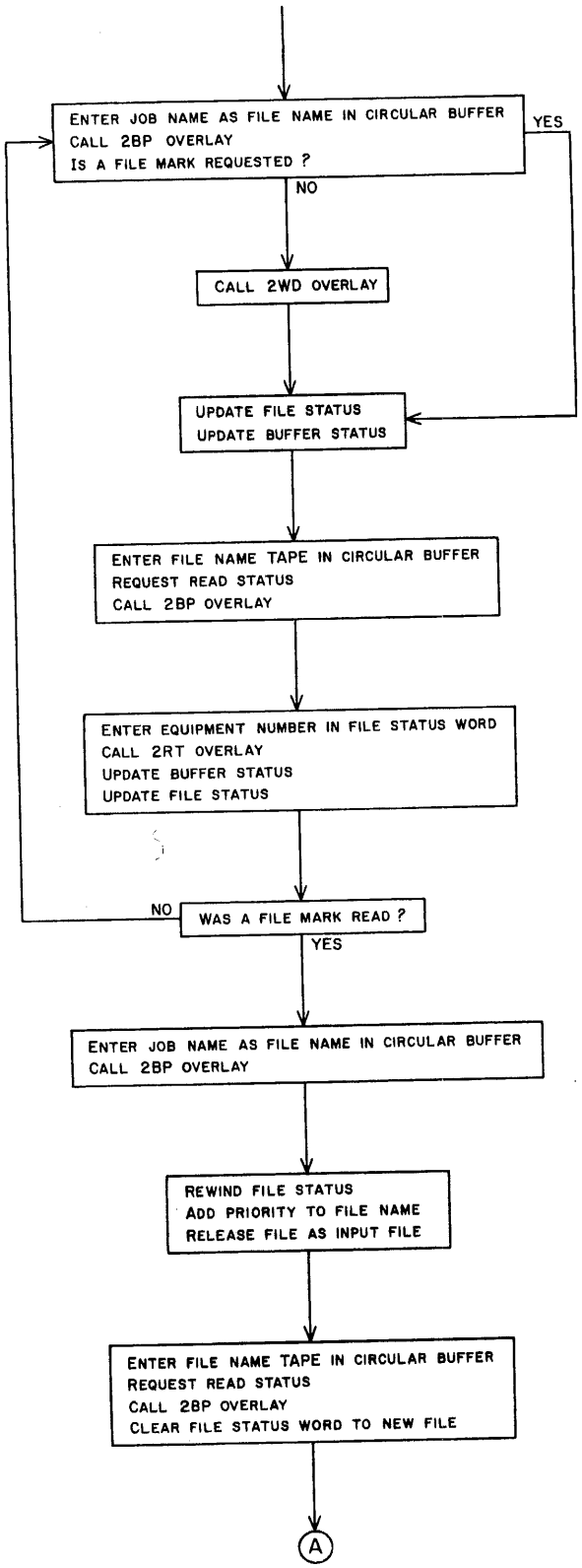
(ILJ CONTINUED)

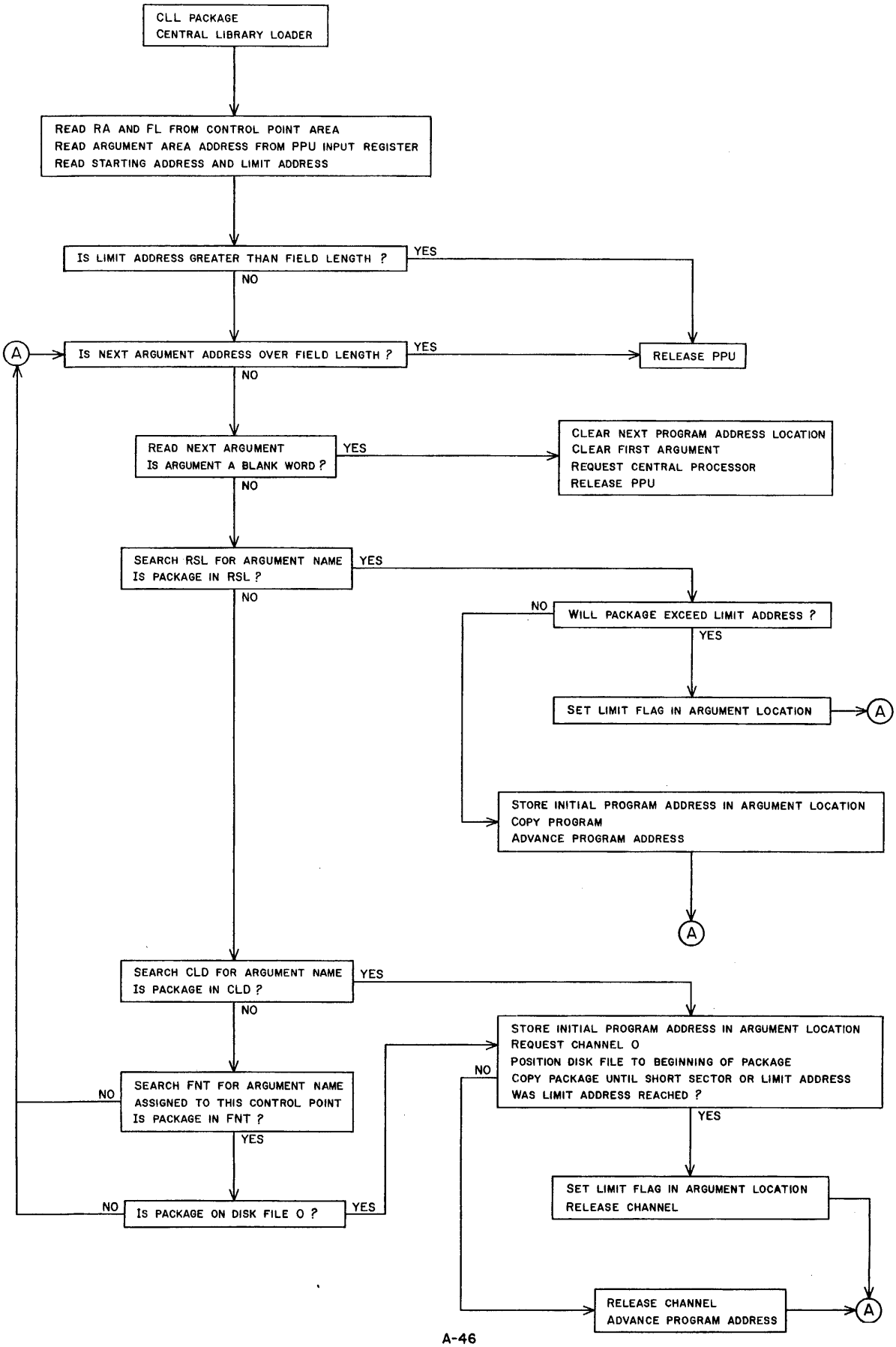


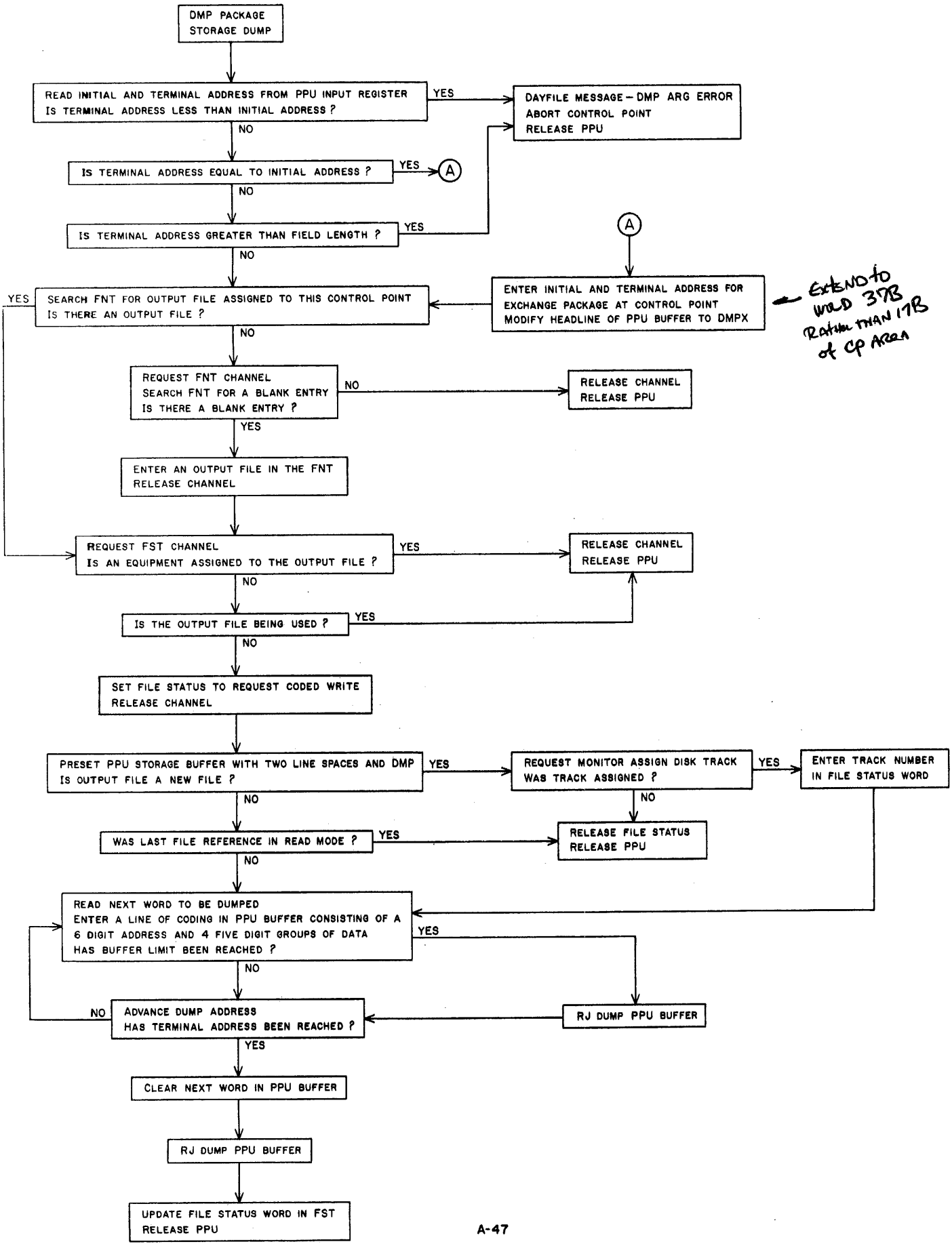

ILT PACKAGE
PHASE ONE TAPE LOAD
DOES CONTROL POINT HAVE A JOB NAME ?
YES
NO
ENTER CONTROL POINT NAME LOAD
REQUEST MONITOR ASSIGN FIELD LENGTH OF 100000B
READ RA AND FL
DOES FL=100000B ?
NO
RELEASE PPU
YES
ENTER BUFFER ADDRESS (0003) IN PPU INPUT REGISTER
CLEAR RA THRU RA + 2
ENTER BUFFER PARAMETERS
CONSOLE MESSAGE – REQUEST TAPE
PAUSE FOR MONITOR
READ RA
IS ERROR FLAG SET ?
YES
RELEASE PPU
NO
NO
HAS OPERATOR ASSIGNED AN EQUIPMENT NUMBER ?
YES
ENTER EQUIPMENT NUMBER IN THIRD BYTE OF PPU INPUT REGISTER
CLEAR OPERATOR ASSIGNMENT IN CONTROL POINT AREA
CLEAR CONSOLE MESSAGE
A
ENTER FILE NAME TAPE IN CIRCULAR BUFFER
REQUEST READ STATUS
CALL 2BP OVERLAY
ENTER EQUIPMENT NUMBER IN FILE STATUS WORD
CALL 2RT OVERLAY
UPDATE BUFFER STATUS
UPDATE FILE STATUS
WAS A FILE MARK READ ?
YES
RELEASE PPU
NO
CALL 2TJ OVERLAY
ENTER NEW JOB NAME IN CONTROL POINT AREA
DAYFILE MESSAGE – LOAD

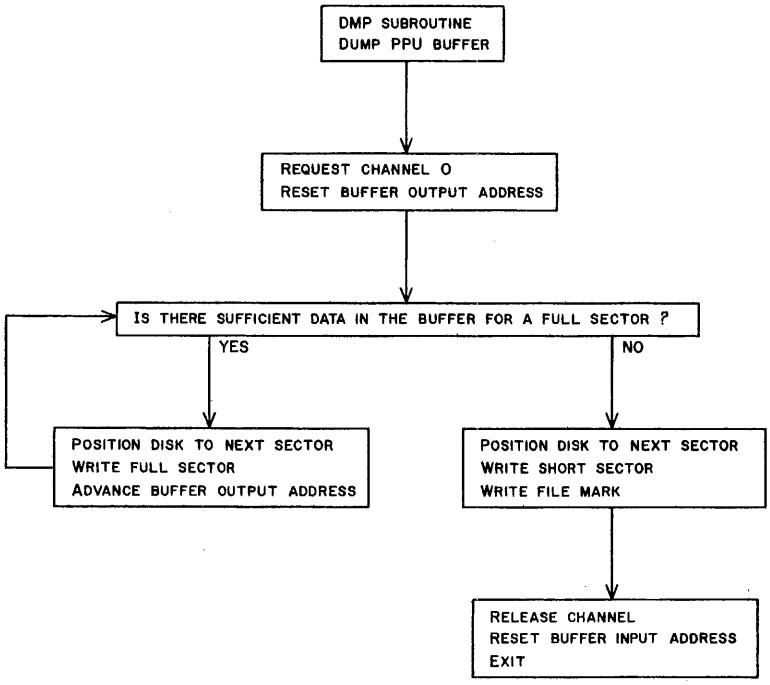
(NEXT PAGE)

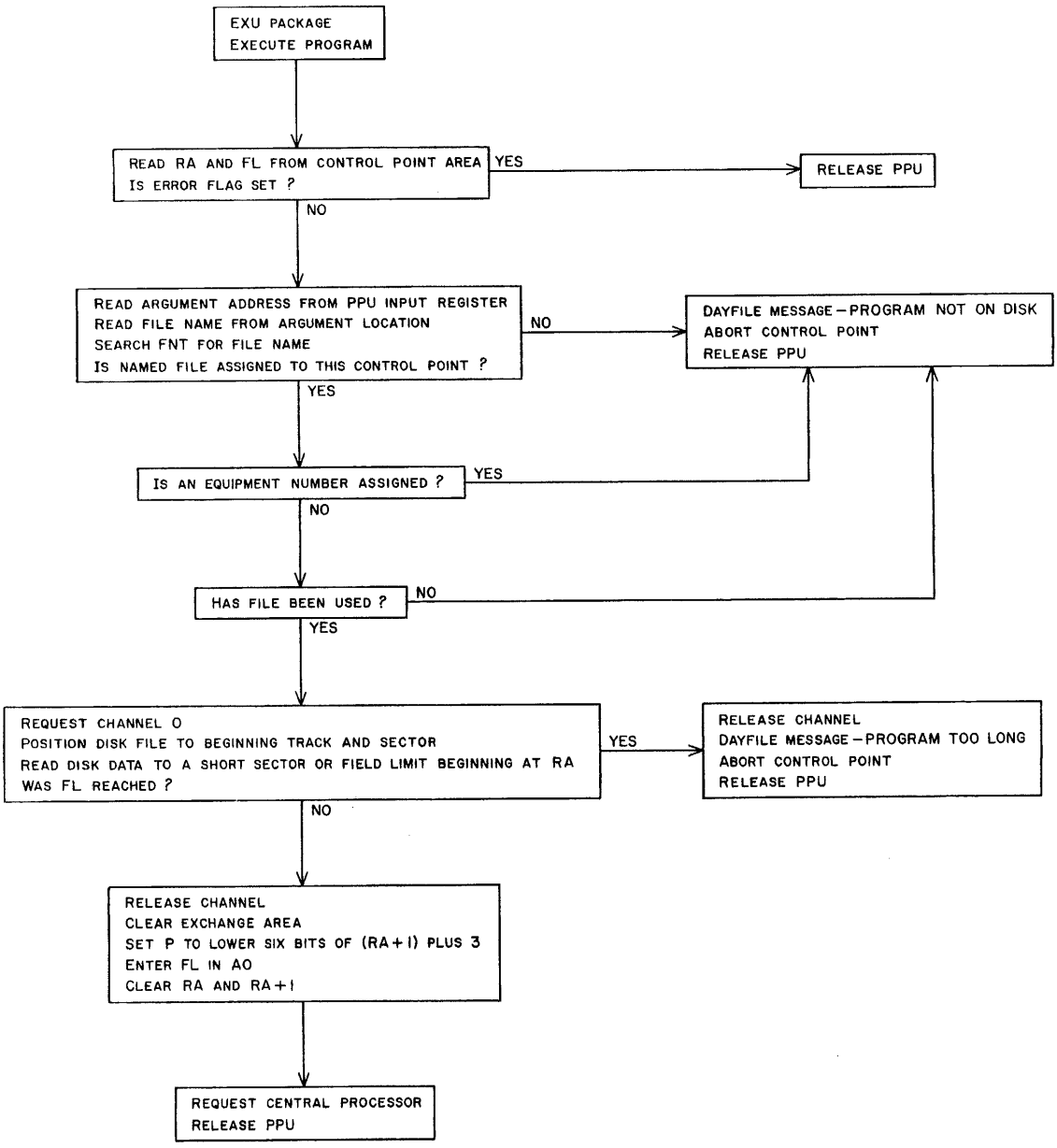
(ILT CONTINUED)

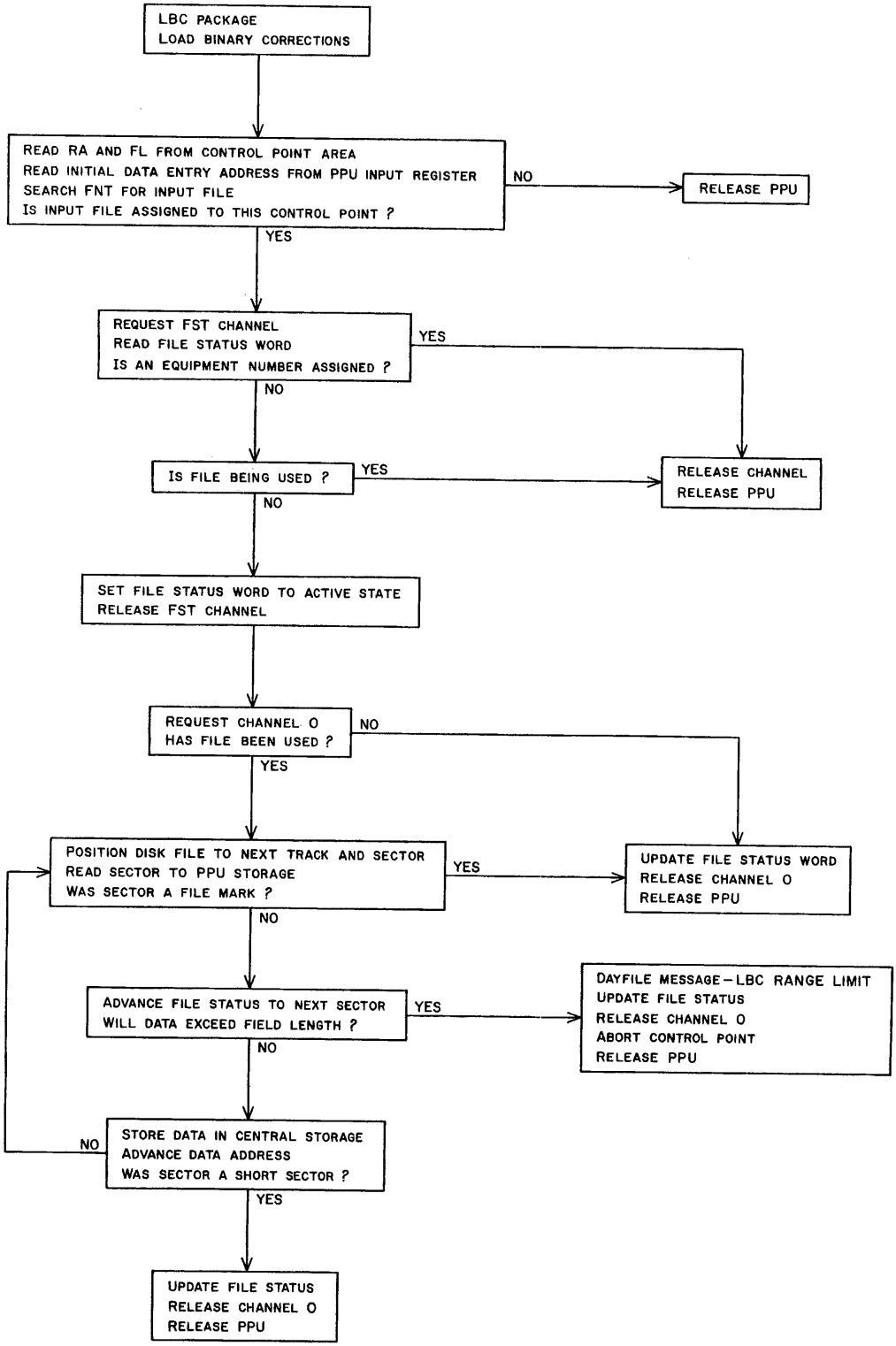


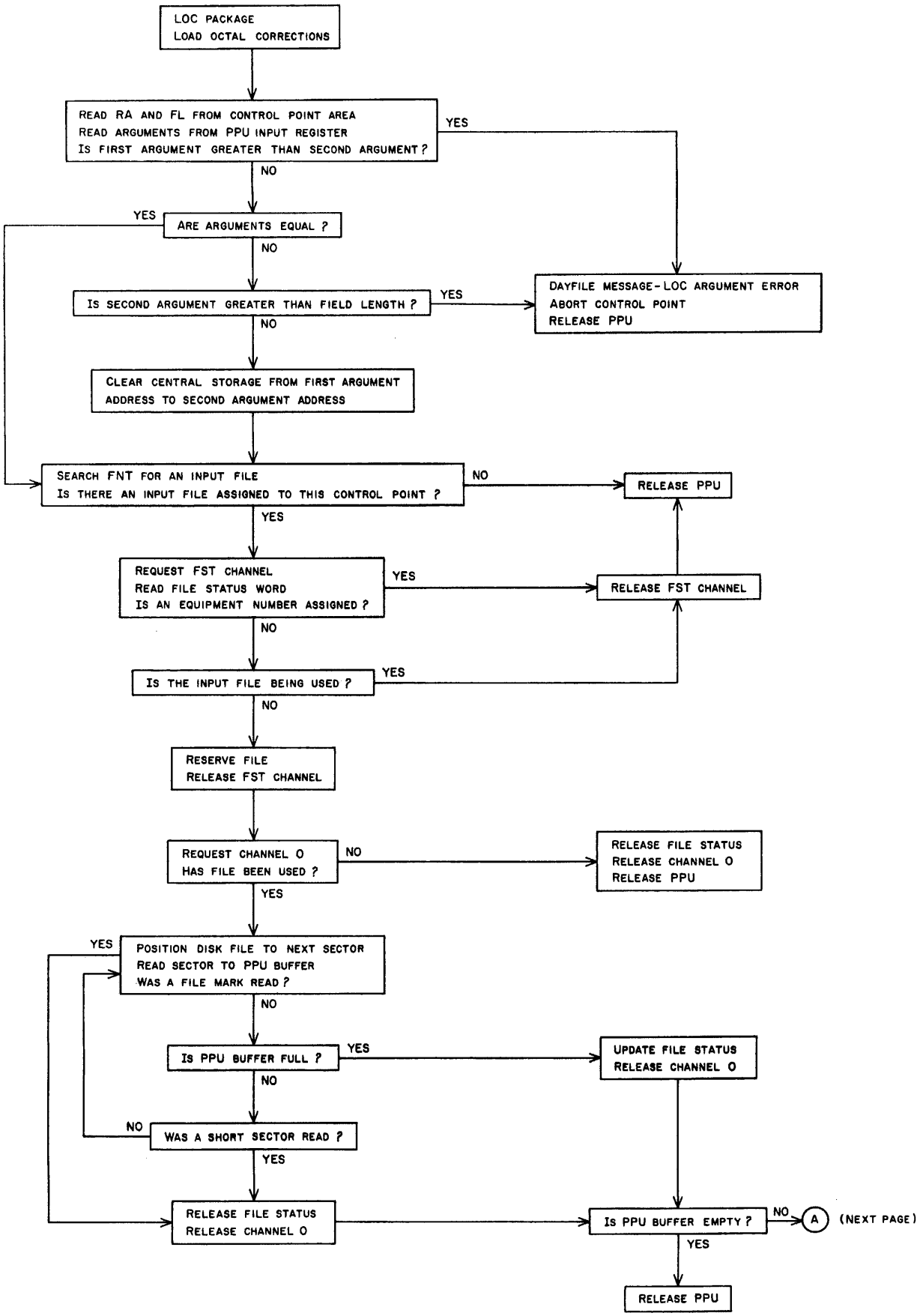

CLL PACKAGE
CENTRAL LIBRARY LOADER
READ RA AND FL FROM CONTROL POINT AREA
READ ARGUMENT AREA ADDRESS FROM PPU INPUT REGISTER
READ STARTING ADDRESS AND LIMIT ADDRESS
IS LIMIT ADDRESS GREATER THAN FIELD LENGTH ?
YES
NO
A
IS NEXT ARGUMENT ADDRESS OVER FIELD LENGTH ?
YES
RELEASE PPU
NO
READ NEXT ARGUMENT
IS ARGUMENT A BLANK WORD ?
YES
CLEAR NEXT PROGRAM ADDRESS LOCATION
CLEAR FIRST ARGUMENT
REQUEST CENTRAL PROCESSOR
RELEASE PPU
NO
SEARCH RSL FOR ARGUMENT NAME
IS PACKAGE IN RSL ?
YES
NO
NO
WILL PACKAGE EXCEED LIMIT ADDRESS ?
YES
SET LIMIT FLAG IN ARGUMENT LOCATION
A
STORE INITIAL PROGRAM ADDRESS IN ARGUMENT LOCATION
COPY PROGRAM
ADVANCE PROGRAM ADDRESS
A
SEARCH CLD FOR ARGUMENT NAME
IS PACKAGE IN CLD ?
YES
NO
STORE INITIAL PROGRAM ADDRESS IN ARGUMENT LOCATION
REQUEST CHANNEL 0
POSITION DISK FILE TO BEGINNING OF PACKAGE
COPY PACKAGE UNTIL SHORT SECTOR OR LIMIT ADDRESS
WAS LIMIT ADDRESS REACHED ?
NO
SEARCH FNT FOR ARGUMENT NAME
ASSIGNED TO THIS CONTROL POINT
IS PACKAGE IN FNT ?
NO
YES
YES
SET LIMIT FLAG IN ARGUMENT LOCATION
RELEASE CHANNEL
NO
IS PACKAGE ON DISK FILE 0 ?
YES
RELEASE CHANNEL
ADVANCE PROGRAM ADDRESS
A

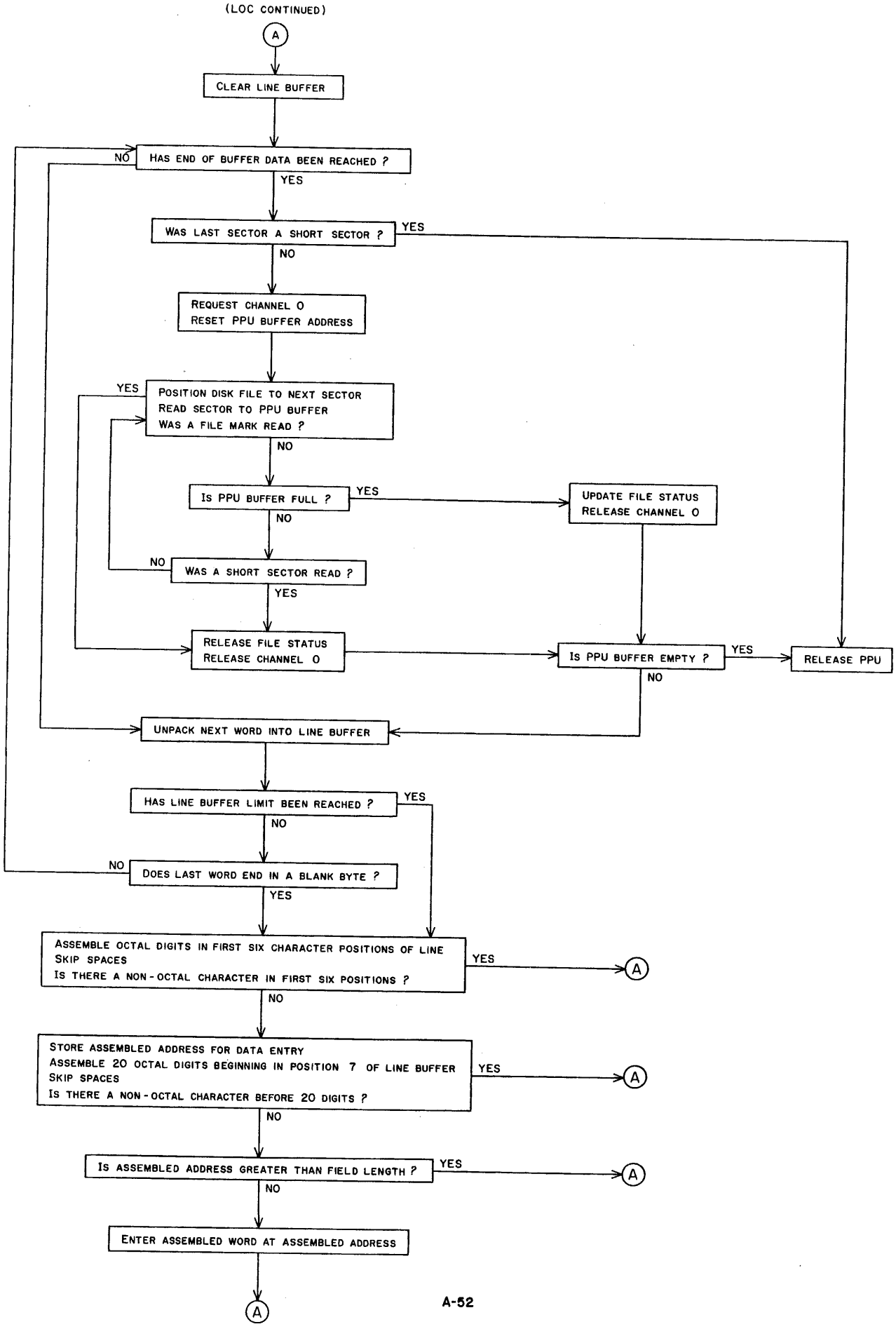

DMP PACKAGE STORAGE DUMP
READ INITIAL AND TERMINAL ADDRESS FROM PPU INPUT REGISTER
IS TERMINAL ADDRESS LESS THAN INITIAL ADDRESS ?
YES
DAYFILE MESSAGE – DMP ARG ERROR
ABORT CONTROL POINT
RELEASE PPU
NO
IS TERMINAL ADDRESS EQUAL TO INITIAL ADDRESS ?
YES
A
NO
IS TERMINAL ADDRESS GREATER THAN FIELD LENGTH ?
YES
NO
A
ENTER INITIAL AND TERMINAL ADDRESS FOR EXCHANGE PACKAGE AT CONTROL POINT
MODIFY HEADLINE OF PPU BUFFER TO DMPX
EXTEND TO WORD 37B RATHER THAN 17B OF CP AREA
YES
SEARCH FNT FOR OUTPUT FILE ASSIGNED TO THIS CONTROL POINT
IS THERE AN OUTPUT FILE ?
NO
REQUEST FNT CHANNEL
SEARCH FNT FOR A BLANK ENTRY
IS THERE A BLANK ENTRY ?
NO
RELEASE CHANNEL
RELEASE PPU
YES
ENTER AN OUTPUT FILE IN THE FNT
RELEASE CHANNEL
REQUEST FST CHANNEL
IS AN EQUIPMENT ASSIGNED TO THE OUTPUT FILE ?
YES
RELEASE CHANNEL
RELEASE PPU
NO
IS THE OUTPUT FILE BEING USED ?
YES
NO
SET FILE STATUS TO REQUEST CODED WRITE
RELEASE CHANNEL
PRESET PPU STORAGE BUFFER WITH TWO LINE SPACES AND DMP
IS OUTPUT FILE A NEW FILE ?
YES
REQUEST MONITOR ASSIGN DISK TRACK
WAS TRACK ASSIGNED ?
YES
ENTER TRACK NUMBER IN FILE STATUS WORD
NO
NO
WAS LAST FILE REFERENCE IN READ MODE ?
YES
RELEASE FILE STATUS
RELEASE PPU
NO
READ NEXT WORD TO BE DUMPED
ENTER A LINE OF CODING IN PPU BUFFER CONSISTING OF A 6 DIGIT ADDRESS AND 4 FIVE DIGIT GROUPS OF DATA
HAS BUFFER LIMIT BEEN REACHED ?
YES
NO
NO
ADVANCE DUMP ADDRESS
HAS TERMINAL ADDRESS BEEN REACHED ?
RJ DUMP PPU BUFFER
YES
CLEAR NEXT WORD IN PPU BUFFER
RJ DUMP PPU BUFFER
UPDATE FILE STATUS WORD IN FST
RELEASE PPU

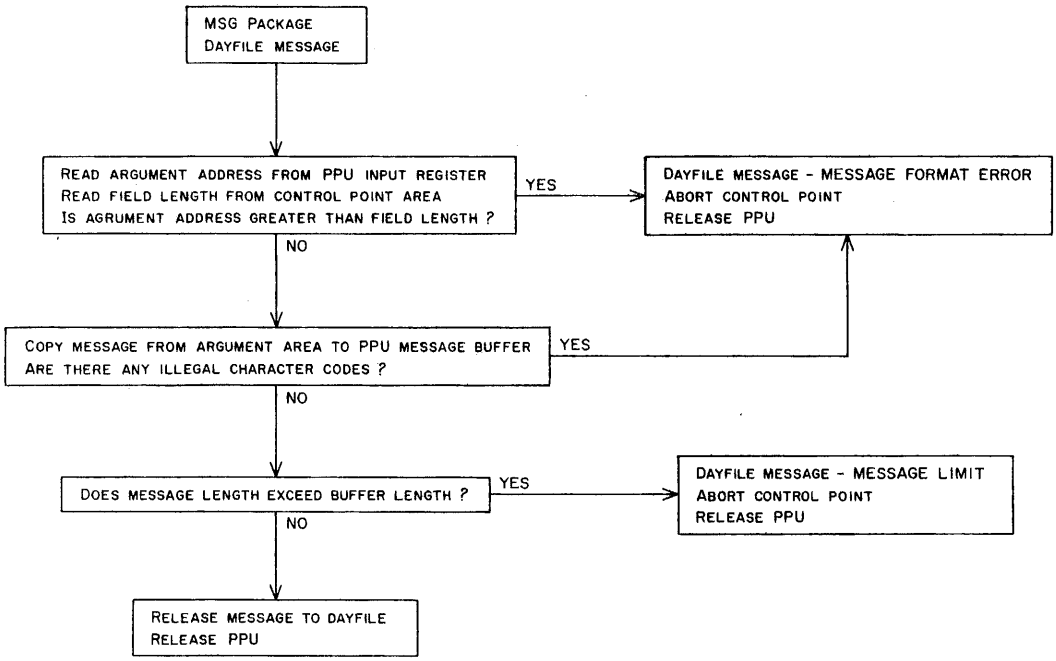

DMP SUBROUTINE
DUMP PPU BUFFER
REQUEST CHANNEL 0
RESET BUFFER OUTPUT ADDRESS
IS THERE SUFFICIENT DATA IN THE BUFFER FOR A FULL SECTOR ?
YES
NO
POSITION DISK TO NEXT SECTOR
WRITE FULL SECTOR
ADVANCE BUFFER OUTPUT ADDRESS
POSITION DISK TO NEXT SECTOR
WRITE SHORT SECTOR
WRITE FILE MARK
RELEASE CHANNEL
RESET BUFFER INPUT ADDRESS
EXIT

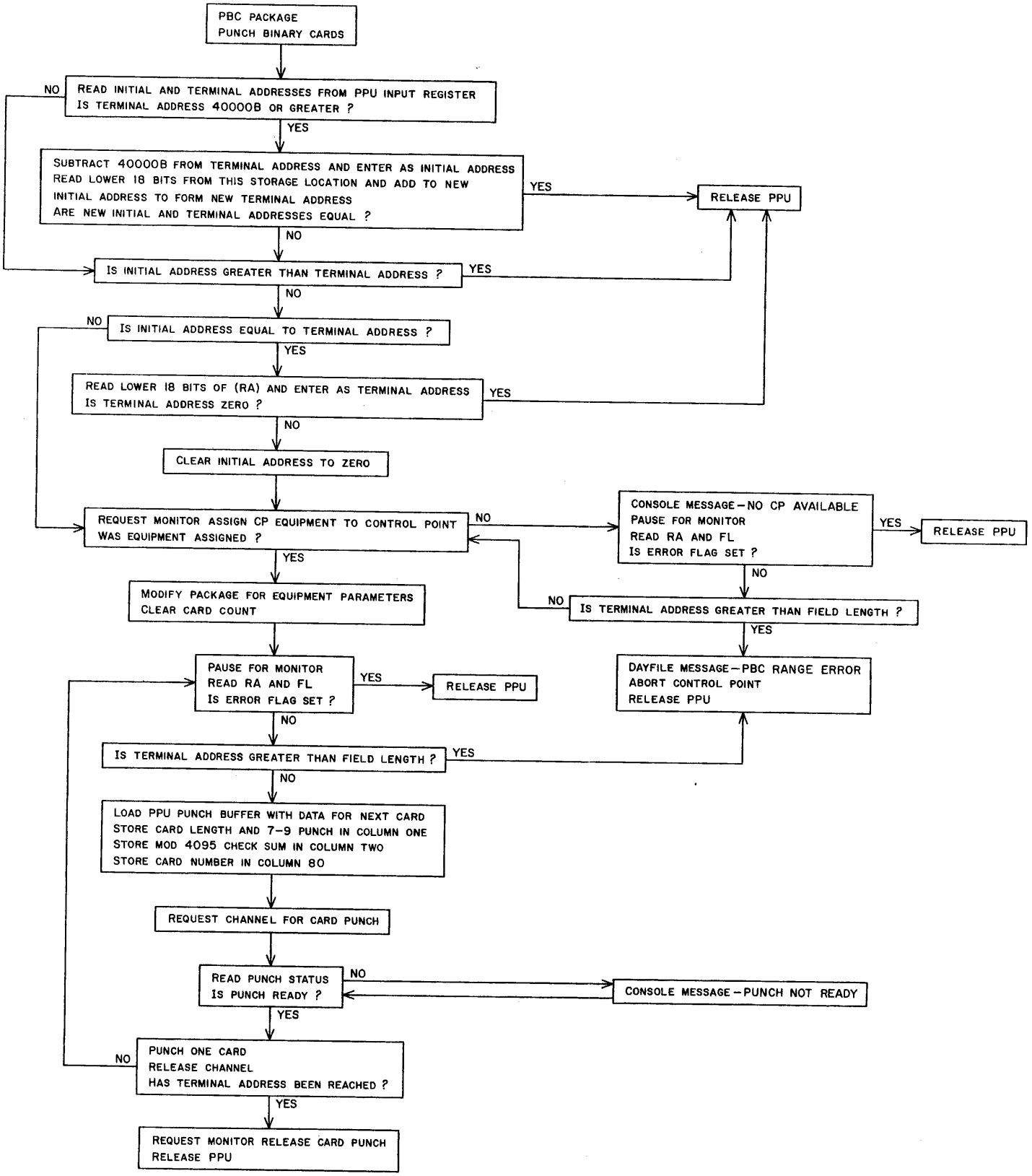

EXU PACKAGE
EXECUTE PROGRAM
READ RA AND FL FROM CONTROL POINT AREA
IS ERROR FLAG SET ?
YES
RELEASE PPU
NO
READ ARGUMENT ADDRESS FROM PPU INPUT REGISTER
READ FILE NAME FROM ARGUMENT LOCATION
SEARCH FNT FOR FILE NAME
IS NAMED FILE ASSIGNED TO THIS CONTROL POINT ?
NO
DAYFILE MESSAGE-PROGRAM NOT ON DISK
ABORT CONTROL POINT
RELEASE PPU
YES
IS AN EQUIPMENT NUMBER ASSIGNED ?
YES
NO
HAS FILE BEEN USED ?
NO
YES
REQUEST CHANNEL 0
POSITION DISK FILE TO BEGINNING TRACK AND SECTOR
READ DISK DATA TO A SHORT SECTOR OR FIELD LIMIT BEGINNING AT RA
WAS FL REACHED ?
YES
RELEASE CHANNEL
DAYFILE MESSAGE-PROGRAM TOO LONG
ABORT CONTROL POINT
RELEASE PPU
NO
RELEASE CHANNEL
CLEAR EXCHANGE AREA
SET P TO LOWER SIX BITS OF (RA+1) PLUS 3
ENTER FL IN A0
CLEAR RA AND RA+1
REQUEST CENTRAL PROCESSOR
RELEASE PPU

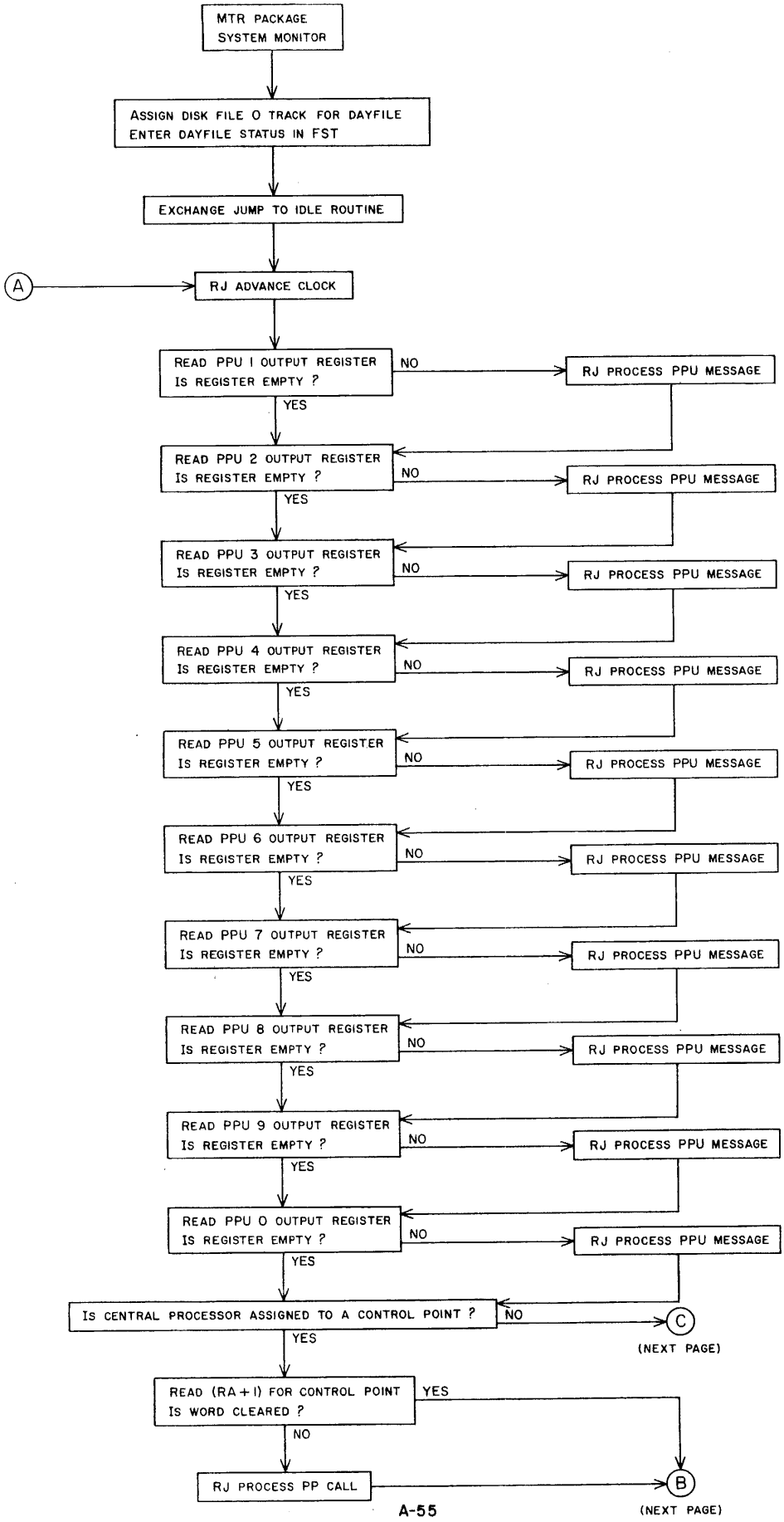

LBC PACKAGE
LOAD BINARY CORRECTIONS
READ RA AND FL FROM CONTROL POINT AREA
READ INITIAL DATA ENTRY ADDRESS FROM PPU INPUT REGISTER
SEARCH FNT FOR INPUT FILE
IS INPUT FILE ASSIGNED TO THIS CONTROL POINT ?
NO
RELEASE PPU
YES
REQUEST FST CHANNEL
READ FILE STATUS WORD
IS AN EQUIPMENT NUMBER ASSIGNED ?
YES
NO
IS FILE BEING USED ?
YES
RELEASE CHANNEL
RELEASE PPU
NO
SET FILE STATUS WORD TO ACTIVE STATE
RELEASE FST CHANNEL
REQUEST CHANNEL 0
HAS FILE BEEN USED ?
NO
YES
POSITION DISK FILE TO NEXT TRACK AND SECTOR
READ SECTOR TO PPU STORAGE
WAS SECTOR A FILE MARK ?
YES
UPDATE FILE STATUS WORD
RELEASE CHANNEL 0
RELEASE PPU
NO
ADVANCE FILE STATUS TO NEXT SECTOR
WILL DATA EXCEED FIELD LENGTH ?
YES
DAYFILE MESSAGE-LBC RANGE LIMIT
UPDATE FILE STATUS
RELEASE CHANNEL 0
ABORT CONTROL POINT
RELEASE PPU
NO
STORE DATA IN CENTRAL STORAGE
ADVANCE DATA ADDRESS
WAS SECTOR A SHORT SECTOR ?
NO
YES
UPDATE FILE STATUS
RELEASE CHANNEL 0
RELEASE PPU

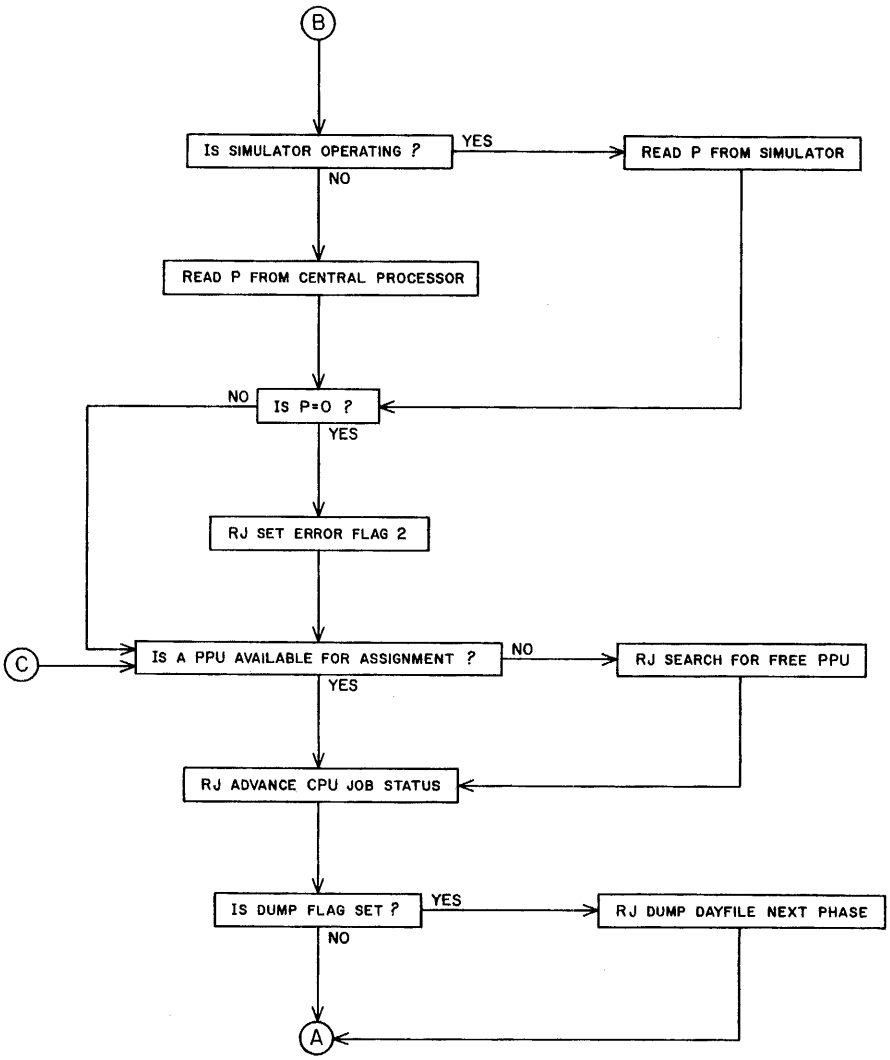

LOC PACKAGE
LOAD OCTAL CORRECTIONS
READ RA AND FL FROM CONTROL POINT AREA
READ ARGUMENTS FROM PPU INPUT REGISTER
IS FIRST ARGUMENT GREATER THAN SECOND ARGUMENT ?
YES
NO
ARE ARGUMENTS EQUAL ?
YES
NO
IS SECOND ARGUMENT GREATER THAN FIELD LENGTH ?
YES
NO
DAYFILE MESSAGE - LOC ARGUMENT ERROR
ABORT CONTROL POINT
RELEASE PPU
CLEAR CENTRAL STORAGE FROM FIRST ARGUMENT
ADDRESS TO SECOND ARGUMENT ADDRESS
SEARCH FNT FOR AN INPUT FILE
IS THERE AN INPUT FILE ASSIGNED TO THIS CONTROL POINT ?
NO
RELEASE PPU
YES
REQUEST FST CHANNEL
READ FILE STATUS WORD
IS AN EQUIPMENT NUMBER ASSIGNED ?
YES
RELEASE FST CHANNEL
NO
IS THE INPUT FILE BEING USED ?
YES
NO
RESERVE FILE
RELEASE FST CHANNEL
REQUEST CHANNEL 0
HAS FILE BEEN USED ?
NO
RELEASE FILE STATUS
RELEASE CHANNEL 0
RELEASE PPU
YES
YES
POSITION DISK FILE TO NEXT SECTOR
READ SECTOR TO PPU BUFFER
WAS A FILE MARK READ ?
NO
IS PPU BUFFER FULL ?
YES
UPDATE FILE STATUS
RELEASE CHANNEL 0
NO
NO
WAS A SHORT SECTOR READ ?
YES
RELEASE FILE STATUS
RELEASE CHANNEL 0
IS PPU BUFFER EMPTY ?
NO
A
(NEXT PAGE)
YES
RELEASE PPU

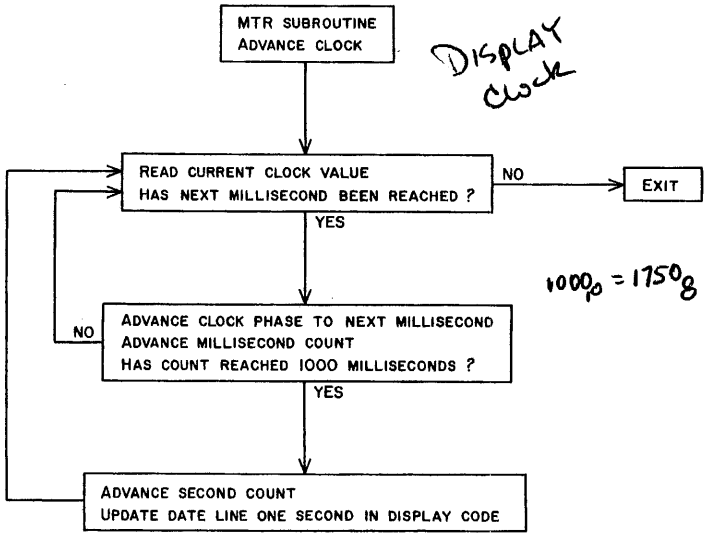

(LOC CONTINUED)
A
CLEAR LINE BUFFER
NO
HAS END OF BUFFER DATA BEEN REACHED ?
YES
WAS LAST SECTOR A SHORT SECTOR ?
YES
NO
REQUEST CHANNEL 0
RESET PPU BUFFER ADDRESS
YES
POSITION DISK FILE TO NEXT SECTOR
READ SECTOR TO PPU BUFFER
WAS A FILE MARK READ ?
NO
IS PPU BUFFER FULL ?
YES
UPDATE FILE STATUS
RELEASE CHANNEL 0
NO
NO
WAS A SHORT SECTOR READ ?
YES
RELEASE FILE STATUS
RELEASE CHANNEL 0
IS PPU BUFFER EMPTY ?
YES
RELEASE PPU
NO
UNPACK NEXT WORD INTO LINE BUFFER
HAS LINE BUFFER LIMIT BEEN REACHED ?
YES
NO
NO
DOES LAST WORD END IN A BLANK BYTE ?
YES
ASSEMBLE OCTAL DIGITS IN FIRST SIX CHARACTER POSITIONS OF LINE
SKIP SPACES
IS THERE A NON-OCTAL CHARACTER IN FIRST SIX POSITIONS ?
YES
A
NO
STORE ASSEMBLED ADDRESS FOR DATA ENTRY
ASSEMBLE 20 OCTAL DIGITS BEGINNING IN POSITION 7 OF LINE BUFFER
SKIP SPACES
IS THERE A NON-OCTAL CHARACTER BEFORE 20 DIGITS ?
YES
A
NO
IS ASSEMBLED ADDRESS GREATER THAN FIELD LENGTH ?
YES
A
NO
ENTER ASSEMBLED WORD AT ASSEMBLED ADDRESS
A

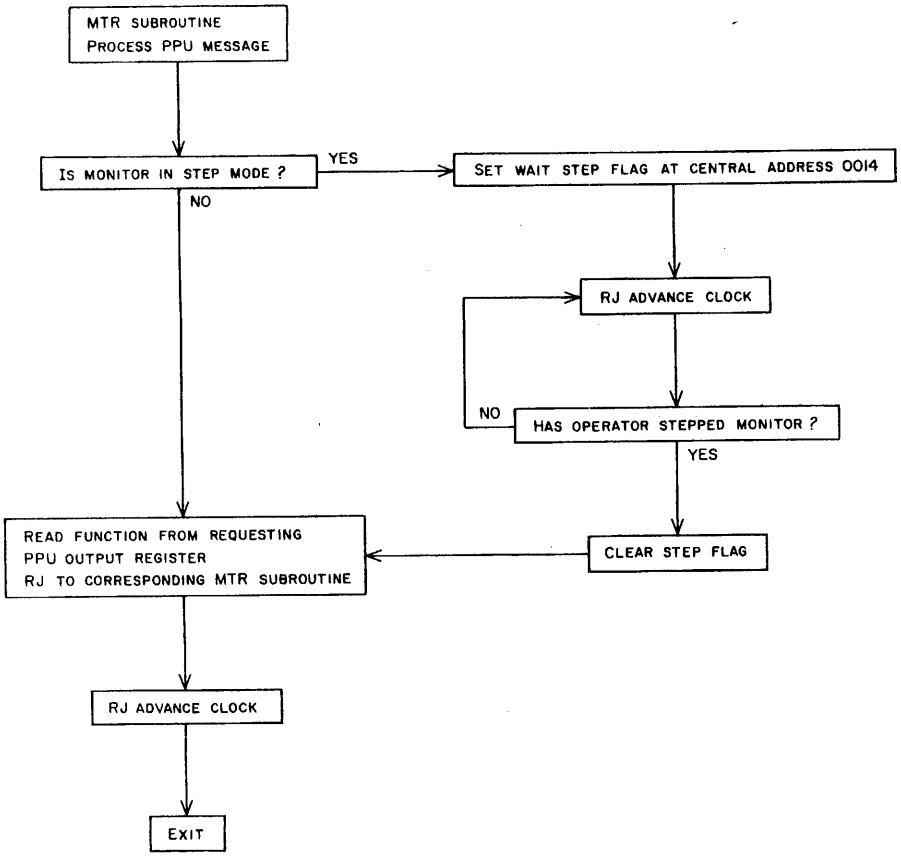

MSG PACKAGE
DAYFILE MESSAGE
READ ARGUMENT ADDRESS FROM PPU INPUT REGISTER
READ FIELD LENGTH FROM CONTROL POINT AREA
IS AGRUMENT ADDRESS GREATER THAN FIELD LENGTH ?
YES
DAYFILE MESSAGE - MESSAGE FORMAT ERROR
ABORT CONTROL POINT
RELEASE PPU
NO
COPY MESSAGE FROM ARGUMENT AREA TO PPU MESSAGE BUFFER
ARE THERE ANY ILLEGAL CHARACTER CODES ?
YES
NO
DOES MESSAGE LENGTH EXCEED BUFFER LENGTH ?
YES
DAYFILE MESSAGE - MESSAGE LIMIT
ABORT CONTROL POINT
RELEASE PPU
NO
RELEASE MESSAGE TO DAYFILE
RELEASE PPU

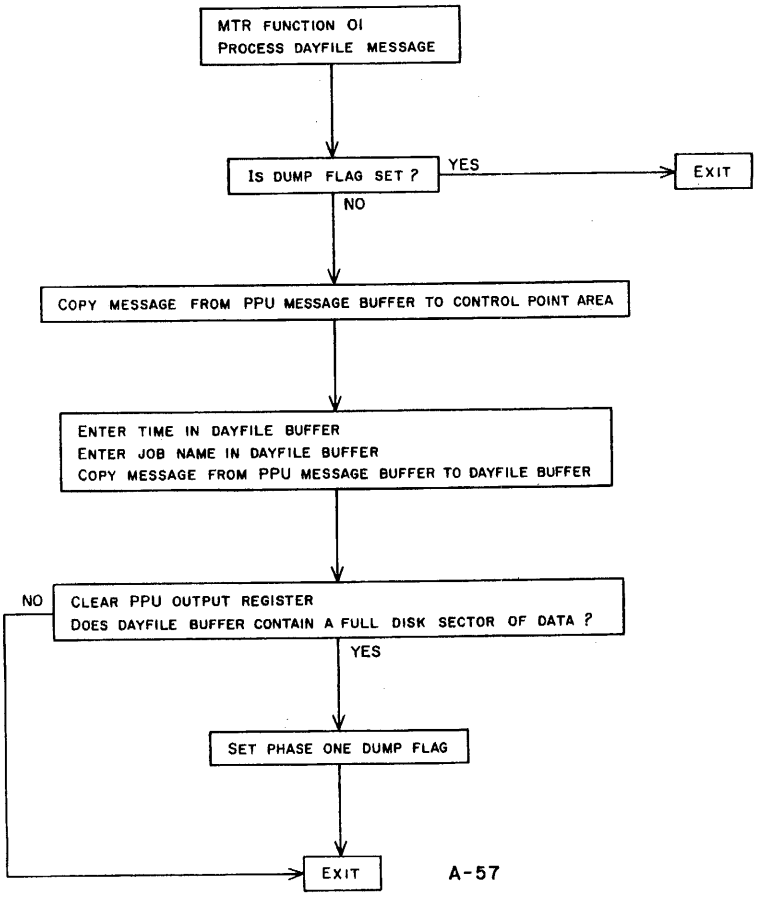

PBC PACKAGE
PUNCH BINARY CARDS
READ INITIAL AND TERMINAL ADDRESSES FROM PPU INPUT REGISTER
IS TERMINAL ADDRESS 40000B OR GREATER ?
NO
YES
SUBTRACT 40000B FROM TERMINAL ADDRESS AND ENTER AS INITIAL ADDRESS
READ LOWER 18 BITS FROM THIS STORAGE LOCATION AND ADD TO NEW
INITIAL ADDRESS TO FORM NEW TERMINAL ADDRESS
ARE NEW INITIAL AND TERMINAL ADDRESSES EQUAL ?
YES
RELEASE PPU
NO
IS INITIAL ADDRESS GREATER THAN TERMINAL ADDRESS ?
YES
NO
NO
IS INITIAL ADDRESS EQUAL TO TERMINAL ADDRESS ?
YES
READ LOWER 18 BITS OF (RA) AND ENTER AS TERMINAL ADDRESS
IS TERMINAL ADDRESS ZERO ?
YES
NO
CLEAR INITIAL ADDRESS TO ZERO
REQUEST MONITOR ASSIGN CP EQUIPMENT TO CONTROL POINT
WAS EQUIPMENT ASSIGNED ?
NO
CONSOLE MESSAGE – NO CP AVAILABLE
PAUSE FOR MONITOR
READ RA AND FL
IS ERROR FLAG SET ?
YES
RELEASE PPU
NO
NO
IS TERMINAL ADDRESS GREATER THAN FIELD LENGTH ?
YES
YES
MODIFY PACKAGE FOR EQUIPMENT PARAMETERS
CLEAR CARD COUNT
PAUSE FOR MONITOR
READ RA AND FL
IS ERROR FLAG SET ?
YES
RELEASE PPU
DAYFILE MESSAGE – PBC RANGE ERROR
ABORT CONTROL POINT
RELEASE PPU
NO
IS TERMINAL ADDRESS GREATER THAN FIELD LENGTH ?
YES
NO
LOAD PPU PUNCH BUFFER WITH DATA FOR NEXT CARD
STORE CARD LENGTH AND 7-9 PUNCH IN COLUMN ONE
STORE MOD 4095 CHECK SUM IN COLUMN TWO
STORE CARD NUMBER IN COLUMN 80
REQUEST CHANNEL FOR CARD PUNCH
READ PUNCH STATUS
IS PUNCH READY ?
NO
CONSOLE MESSAGE – PUNCH NOT READY
YES
NO
PUNCH ONE CARD
RELEASE CHANNEL
HAS TERMINAL ADDRESS BEEN REACHED ?
YES
REQUEST MONITOR RELEASE CARD PUNCH
RELEASE PPU

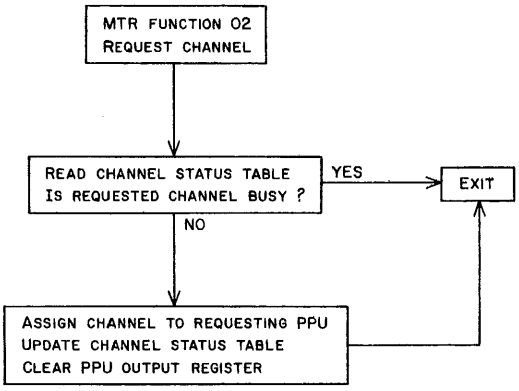

MTR PACKAGE
SYSTEM MONITOR
ASSIGN DISK FILE 0 TRACK FOR DAYFILE
ENTER DAYFILE STATUS IN FST
EXCHANGE JUMP TO IDLE ROUTINE
A
RJ ADVANCE CLOCK
READ PPU 1 OUTPUT REGISTER
IS REGISTER EMPTY ?
NO
RJ PROCESS PPU MESSAGE
YES
READ PPU 2 OUTPUT REGISTER
IS REGISTER EMPTY ?
NO
RJ PROCESS PPU MESSAGE
YES
READ PPU 3 OUTPUT REGISTER
IS REGISTER EMPTY ?
NO
RJ PROCESS PPU MESSAGE
YES
READ PPU 4 OUTPUT REGISTER
IS REGISTER EMPTY ?
NO
RJ PROCESS PPU MESSAGE
YES
READ PPU 5 OUTPUT REGISTER
IS REGISTER EMPTY ?
NO
RJ PROCESS PPU MESSAGE
YES
READ PPU 6 OUTPUT REGISTER
IS REGISTER EMPTY ?
NO
RJ PROCESS PPU MESSAGE
YES
READ PPU 7 OUTPUT REGISTER
IS REGISTER EMPTY ?
NO
RJ PROCESS PPU MESSAGE
YES
READ PPU 8 OUTPUT REGISTER
IS REGISTER EMPTY ?
NO
RJ PROCESS PPU MESSAGE
YES
READ PPU 9 OUTPUT REGISTER
IS REGISTER EMPTY ?
NO
RJ PROCESS PPU MESSAGE
YES
READ PPU 0 OUTPUT REGISTER
IS REGISTER EMPTY ?
NO
RJ PROCESS PPU MESSAGE
YES
IS CENTRAL PROCESSOR ASSIGNED TO A CONTROL POINT ?
NO
C
(NEXT PAGE)
YES
READ (RA+1) FOR CONTROL POINT
IS WORD CLEARED ?
YES
NO
RJ PROCESS PP CALL
B
(NEXT PAGE)

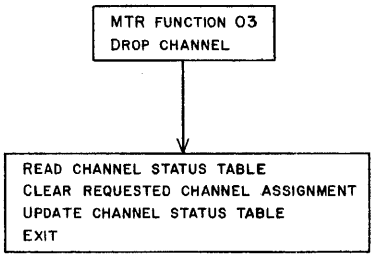

B
IS SIMULATOR OPERATING ?
YES
READ P FROM SIMULATOR
NO
READ P FROM CENTRAL PROCESSOR
NO
IS P=0 ?
YES
RJ SET ERROR FLAG 2
C
IS A PPU AVAILABLE FOR ASSIGNMENT ?
NO
RJ SEARCH FOR FREE PPU
YES
RJ ADVANCE CPU JOB STATUS
IS DUMP FLAG SET ?
YES
RJ DUMP DAYFILE NEXT PHASE
NO
A

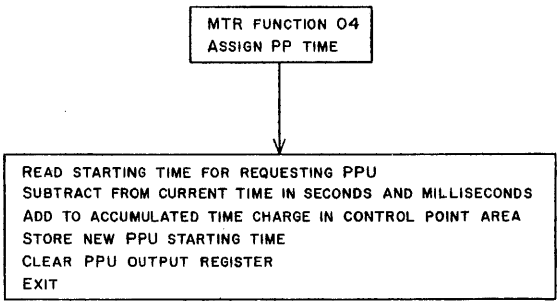

MTR SUBROUTINE
ADVANCE CLOCK
DISPLAY clock
READ CURRENT CLOCK VALUE
HAS NEXT MILLISECOND BEEN REACHED ?
NO
EXIT
YES
1000₁₀ = 1750₈
NO
ADVANCE CLOCK PHASE TO NEXT MILLISECOND
ADVANCE MILLISECOND COUNT
HAS COUNT REACHED 1000 MILLISECONDS ?
YES
ADVANCE SECOND COUNT
UPDATE DATE LINE ONE SECOND IN DISPLAY CODE

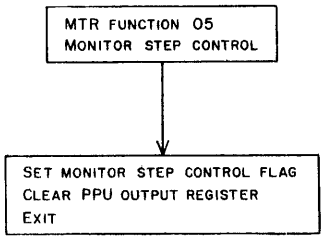

MTR SUBROUTINE
PROCESS PPU MESSAGE
IS MONITOR IN STEP MODE ?
YES
NO
SET WAIT STEP FLAG AT CENTRAL ADDRESS 0014
RJ ADVANCE CLOCK
NO
HAS OPERATOR STEPPED MONITOR ?
YES
CLEAR STEP FLAG
READ FUNCTION FROM REQUESTING
PPU OUTPUT REGISTER
RJ TO CORRESPONDING MTR SUBROUTINE
RJ ADVANCE CLOCK
EXIT

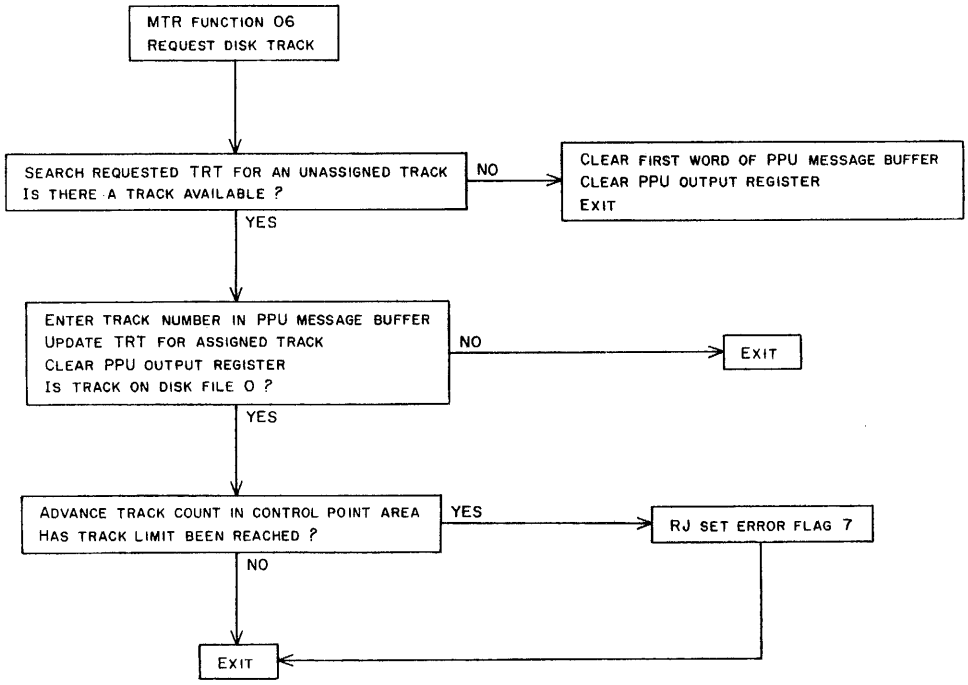

MTR FUNCTION 01
PROCESS DAYFILE MESSAGE
IS DUMP FLAG SET ?
YES
EXIT
NO
COPY MESSAGE FROM PPU MESSAGE BUFFER TO CONTROL POINT AREA
ENTER TIME IN DAYFILE BUFFER
ENTER JOB NAME IN DAYFILE BUFFER
COPY MESSAGE FROM PPU MESSAGE BUFFER TO DAYFILE BUFFER
NO
CLEAR PPU OUTPUT REGISTER
DOES DAYFILE BUFFER CONTAIN A FULL DISK SECTOR OF DATA ?
YES
SET PHASE ONE DUMP FLAG
EXIT

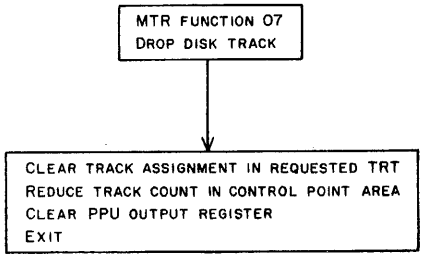

MTR FUNCTION 02
REQUEST CHANNEL
READ CHANNEL STATUS TABLE
IS REQUESTED CHANNEL BUSY ?
YES
EXIT
NO
ASSIGN CHANNEL TO REQUESTING PPU
UPDATE CHANNEL STATUS TABLE
CLEAR PPU OUTPUT REGISTER

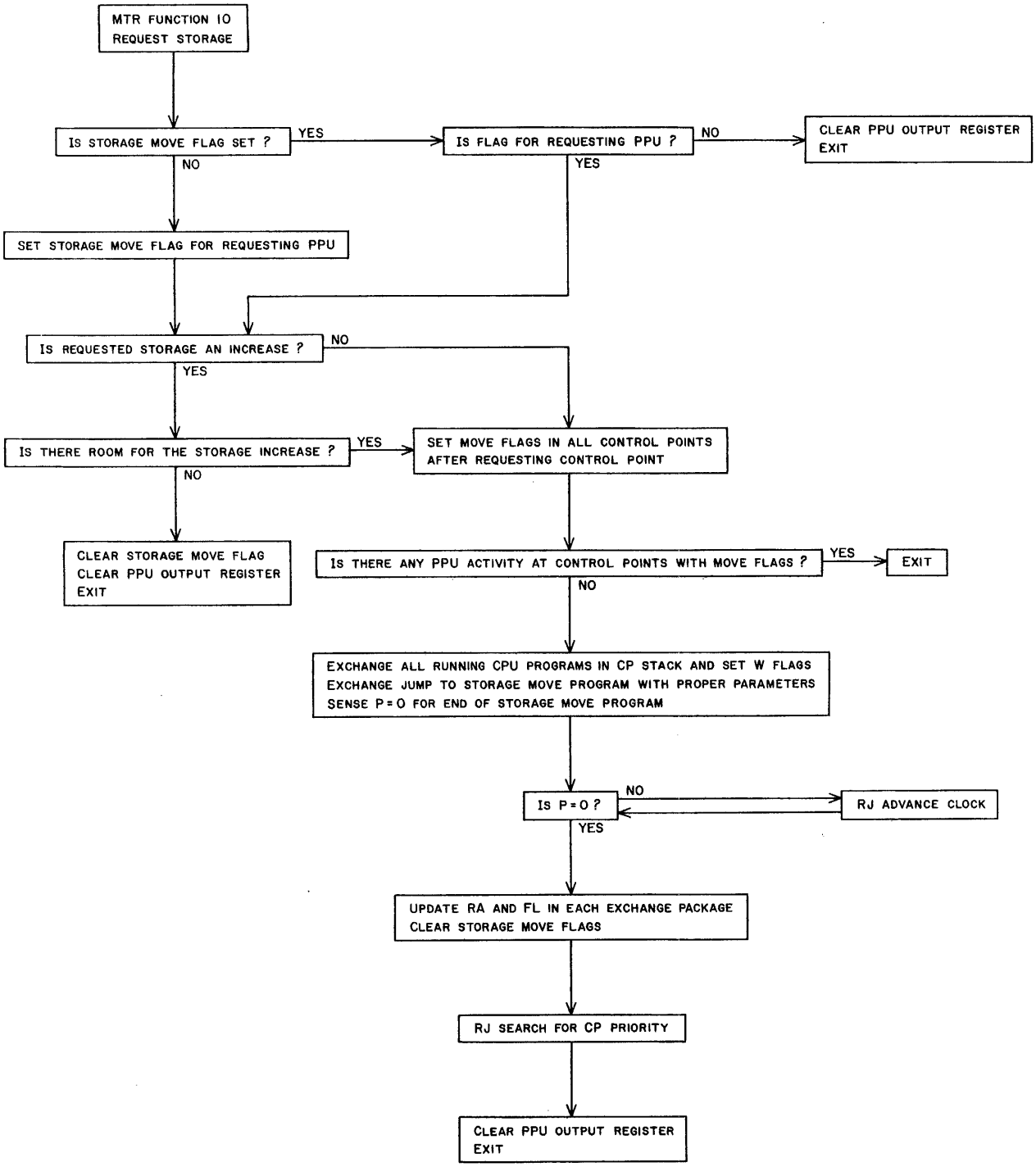

MTR FUNCTION 03
DROP CHANNEL
READ CHANNEL STATUS TABLE
CLEAR REQUESTED CHANNEL ASSIGNMENT
UPDATE CHANNEL STATUS TABLE
EXIT

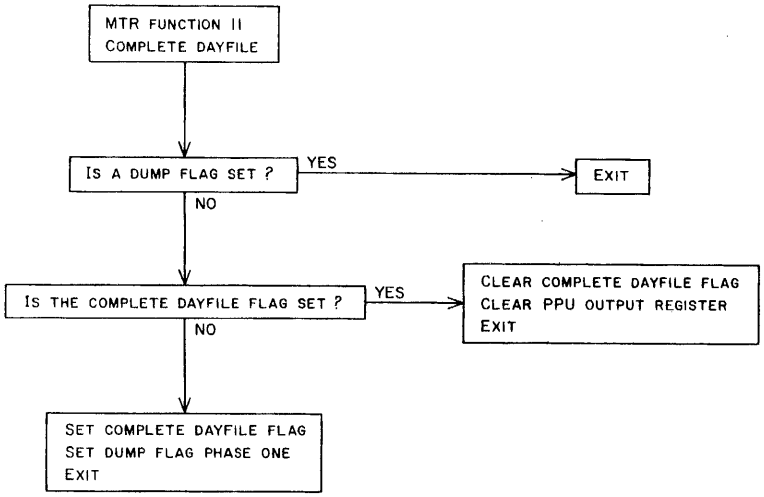

MTR FUNCTION 04
ASSIGN PP TIME
READ STARTING TIME FOR REQUESTING PPU
SUBTRACT FROM CURRENT TIME IN SECONDS AND MILLISECONDS
ADD TO ACCUMULATED TIME CHARGE IN CONTROL POINT AREA
STORE NEW PPU STARTING TIME
CLEAR PPU OUTPUT REGISTER
EXIT

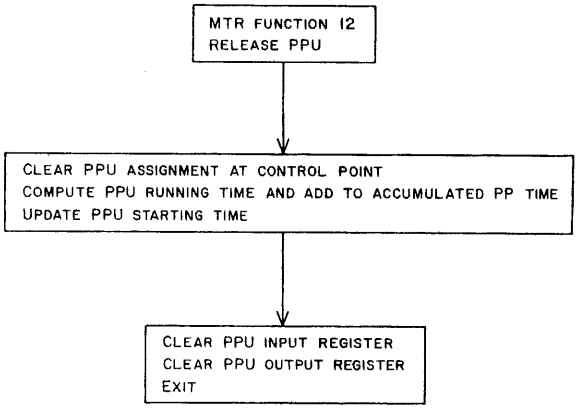

MTR FUNCTION 05
MONITOR STEP CONTROL
SET MONITOR STEP CONTROL FLAG
CLEAR PPU OUTPUT REGISTER
EXIT

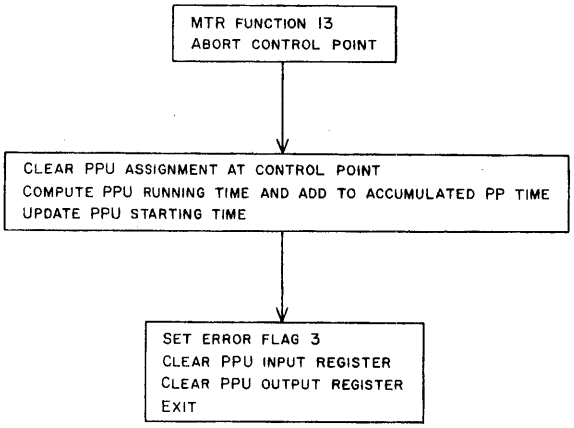

MTR FUNCTION 06
REQUEST DISK TRACK
SEARCH REQUESTED TRT FOR AN UNASSIGNED TRACK
IS THERE A TRACK AVAILABLE ?
NO
CLEAR FIRST WORD OF PPU MESSAGE BUFFER
CLEAR PPU OUTPUT REGISTER
EXIT
YES
ENTER TRACK NUMBER IN PPU MESSAGE BUFFER
UPDATE TRT FOR ASSIGNED TRACK
CLEAR PPU OUTPUT REGISTER
IS TRACK ON DISK FILE 0 ?
NO
EXIT
YES
ADVANCE TRACK COUNT IN CONTROL POINT AREA
HAS TRACK LIMIT BEEN REACHED ?
YES
RJ SET ERROR FLAG 7
NO
EXIT

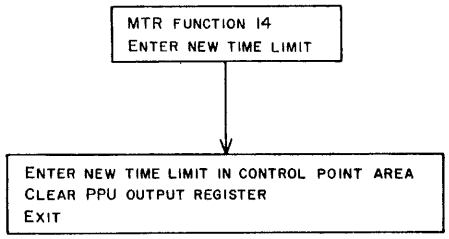

MTR FUNCTION 07
DROP DISK TRACK
CLEAR TRACK ASSIGNMENT IN REQUESTED TRT
REDUCE TRACK COUNT IN CONTROL POINT AREA
CLEAR PPU OUTPUT REGISTER
EXIT

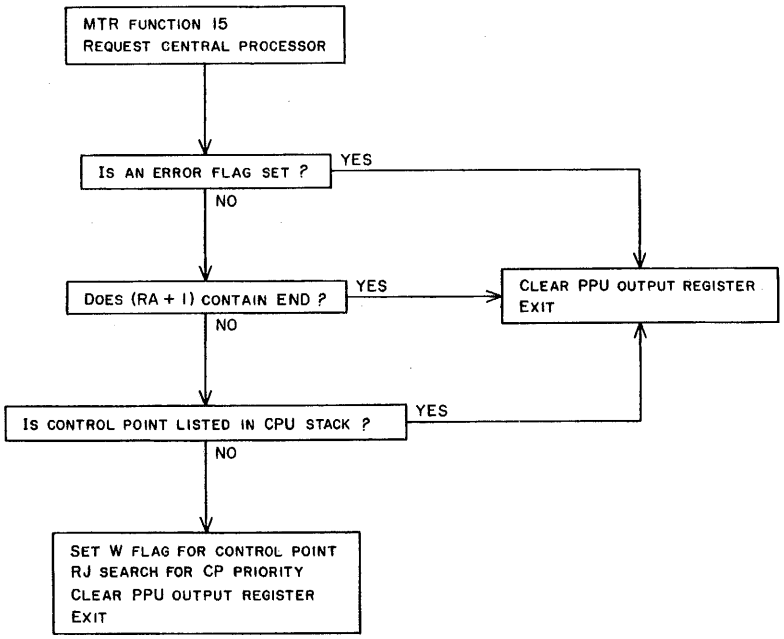

MTR FUNCTION 10
REQUEST STORAGE
IS STORAGE MOVE FLAG SET ?
YES
IS FLAG FOR REQUESTING PPU ?
NO
CLEAR PPU OUTPUT REGISTER
EXIT
NO
YES
SET STORAGE MOVE FLAG FOR REQUESTING PPU
IS REQUESTED STORAGE AN INCREASE ?
NO
YES
IS THERE ROOM FOR THE STORAGE INCREASE ?
YES
SET MOVE FLAGS IN ALL CONTROL POINTS
AFTER REQUESTING CONTROL POINT
NO
CLEAR STORAGE MOVE FLAG
CLEAR PPU OUTPUT REGISTER
EXIT
IS THERE ANY PPU ACTIVITY AT CONTROL POINTS WITH MOVE FLAGS ?
YES
EXIT
NO
EXCHANGE ALL RUNNING CPU PROGRAMS IN CP STACK AND SET W FLAGS
EXCHANGE JUMP TO STORAGE MOVE PROGRAM WITH PROPER PARAMETERS
SENSE P=0 FOR END OF STORAGE MOVE PROGRAM
IS P=0 ?
NO
RJ ADVANCE CLOCK
YES
UPDATE RA AND FL IN EACH EXCHANGE PACKAGE
CLEAR STORAGE MOVE FLAGS
RJ SEARCH FOR CP PRIORITY
CLEAR PPU OUTPUT REGISTER
EXIT

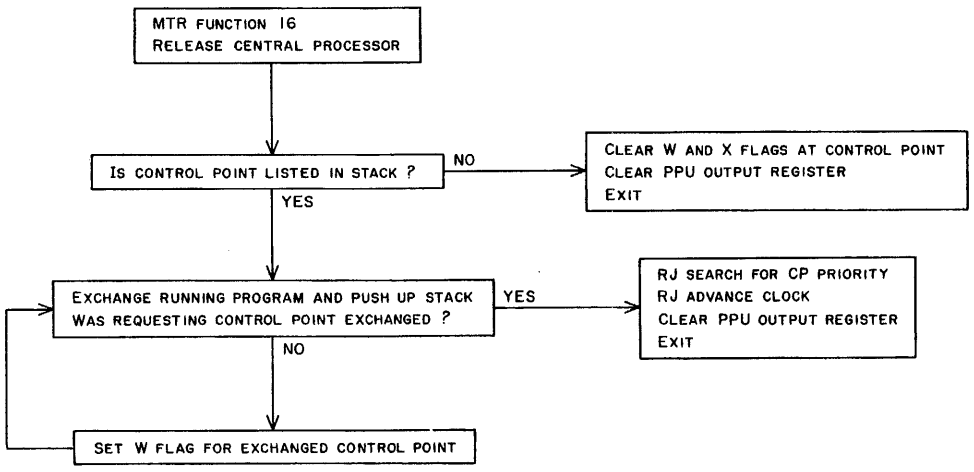

MTR FUNCTION 11
COMPLETE DAYFILE
IS A DUMP FLAG SET ?
YES
EXIT
NO
IS THE COMPLETE DAYFILE FLAG SET ?
YES
CLEAR COMPLETE DAYFILE FLAG
CLEAR PPU OUTPUT REGISTER
EXIT
NO
SET COMPLETE DAYFILE FLAG
SET DUMP FLAG PHASE ONE
EXIT

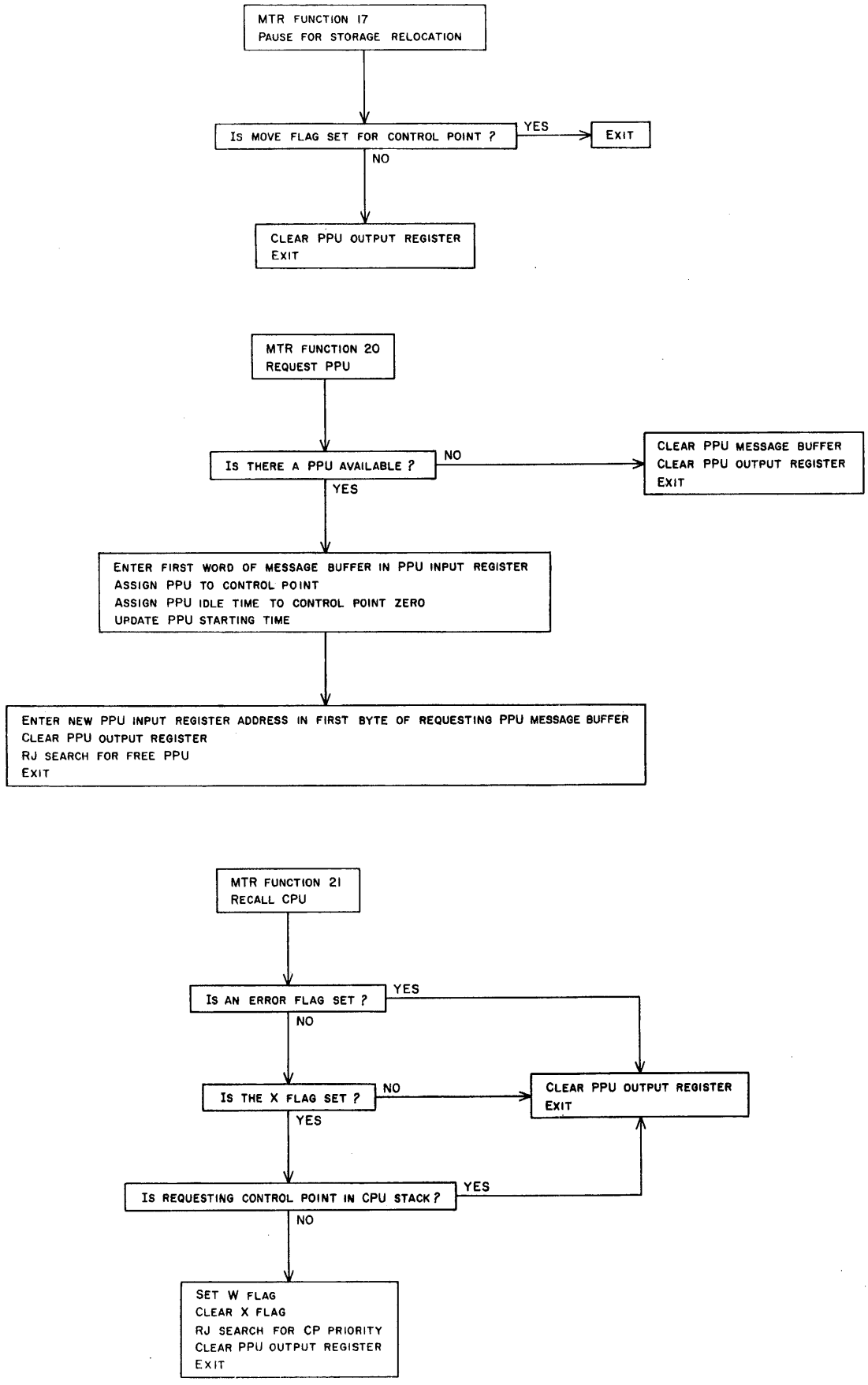

MTR FUNCTION 12
RELEASE PPU
CLEAR PPU ASSIGNMENT AT CONTROL POINT
COMPUTE PPU RUNNING TIME AND ADD TO ACCUMULATED PP TIME
UPDATE PPU STARTING TIME
CLEAR PPU INPUT REGISTER
CLEAR PPU OUTPUT REGISTER
EXIT

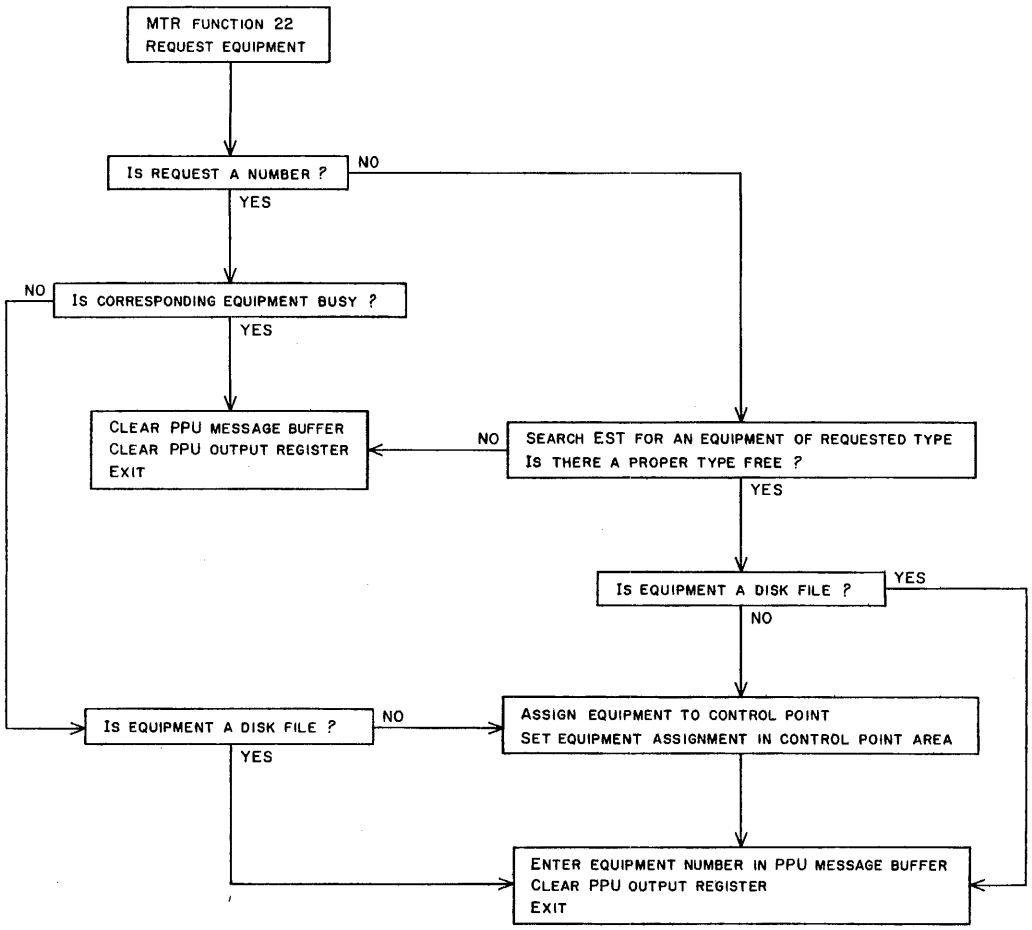

MTR FUNCTION 13
ABORT CONTROL POINT
CLEAR PPU ASSIGNMENT AT CONTROL POINT
COMPUTE PPU RUNNING TIME AND ADD TO ACCUMULATED PP TIME
UPDATE PPU STARTING TIME
SET ERROR FLAG 3
CLEAR PPU INPUT REGISTER
CLEAR PPU OUTPUT REGISTER
EXIT

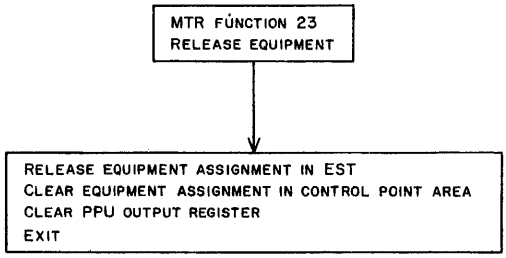

MTR FUNCTION 14
ENTER NEW TIME LIMIT
ENTER NEW TIME LIMIT IN CONTROL POINT AREA
CLEAR PPU OUTPUT REGISTER
EXIT

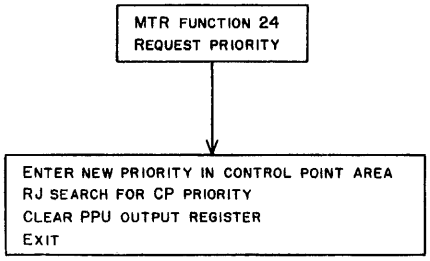

MTR FUNCTION 15
REQUEST CENTRAL PROCESSOR
IS AN ERROR FLAG SET ?
YES
NO
DOES (RA + 1) CONTAIN END ?
YES
NO
CLEAR PPU OUTPUT REGISTER
EXIT
IS CONTROL POINT LISTED IN CPU STACK ?
YES
NO
SET W FLAG FOR CONTROL POINT
RJ SEARCH FOR CP PRIORITY
CLEAR PPU OUTPUT REGISTER
EXIT

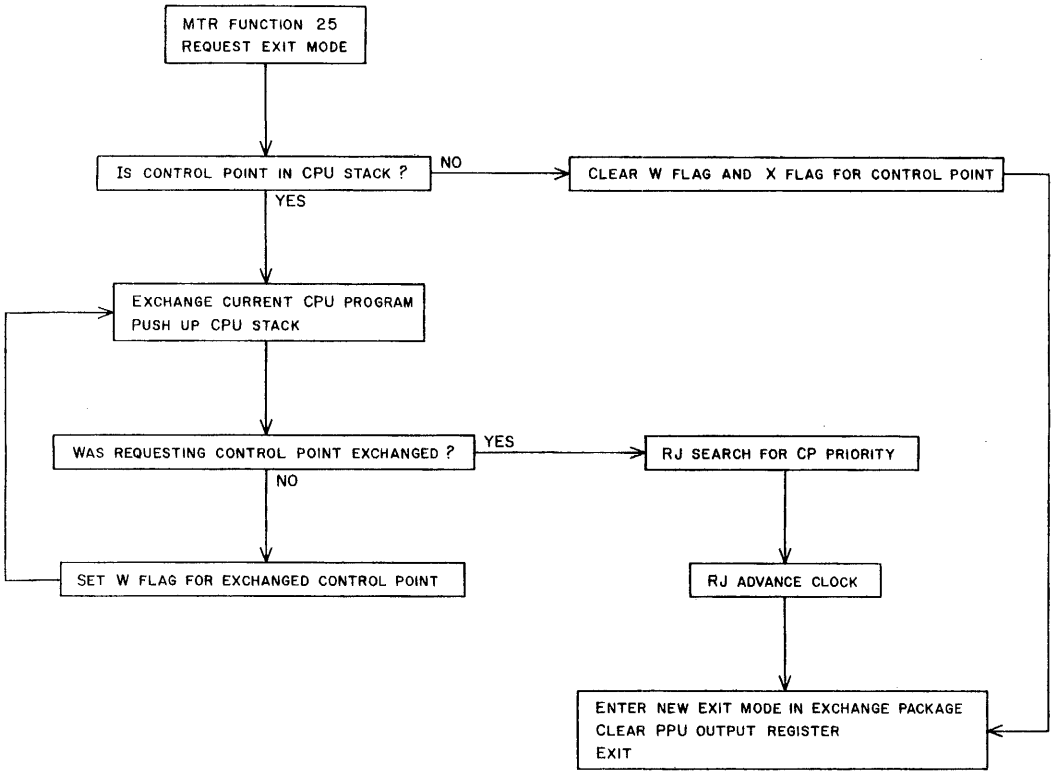

MTR FUNCTION 16
RELEASE CENTRAL PROCESSOR
IS CONTROL POINT LISTED IN STACK ?
NO
YES
CLEAR W AND X FLAGS AT CONTROL POINT
CLEAR PPU OUTPUT REGISTER
EXIT
EXCHANGE RUNNING PROGRAM AND PUSH UP STACK
WAS REQUESTING CONTROL POINT EXCHANGED ?
YES
NO
RJ SEARCH FOR CP PRIORITY
RJ ADVANCE CLOCK
CLEAR PPU OUTPUT REGISTER
EXIT
SET W FLAG FOR EXCHANGED CONTROL POINT

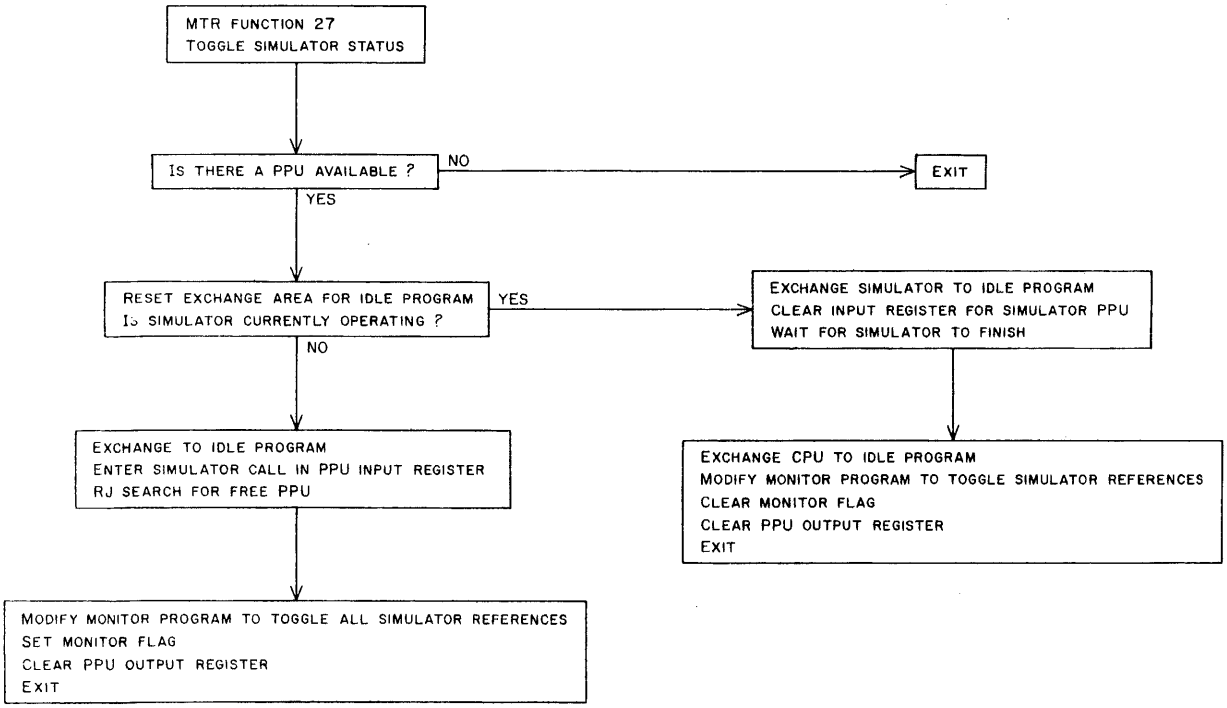

MTR FUNCTION 17
PAUSE FOR STORAGE RELOCATION
IS MOVE FLAG SET FOR CONTROL POINT ?
YES
EXIT
NO
CLEAR PPU OUTPUT REGISTER
EXIT
MTR FUNCTION 20
REQUEST PPU
IS THERE A PPU AVAILABLE ?
NO
CLEAR PPU MESSAGE BUFFER
CLEAR PPU OUTPUT REGISTER
EXIT
YES
ENTER FIRST WORD OF MESSAGE BUFFER IN PPU INPUT REGISTER
ASSIGN PPU TO CONTROL POINT
ASSIGN PPU IDLE TIME TO CONTROL POINT ZERO
UPDATE PPU STARTING TIME
ENTER NEW PPU INPUT REGISTER ADDRESS IN FIRST BYTE OF REQUESTING PPU MESSAGE BUFFER
CLEAR PPU OUTPUT REGISTER
RJ SEARCH FOR FREE PPU
EXIT
MTR FUNCTION 21
RECALL CPU
IS AN ERROR FLAG SET ?
YES
NO
IS THE X FLAG SET ?
NO
CLEAR PPU OUTPUT REGISTER
EXIT
YES
IS REQUESTING CONTROL POINT IN CPU STACK ?
YES
NO
SET W FLAG
CLEAR X FLAG
RJ SEARCH FOR CP PRIORITY
CLEAR PPU OUTPUT REGISTER
EXIT

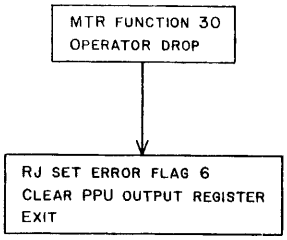

MTR FUNCTION 22
REQUEST EQUIPMENT
IS REQUEST A NUMBER ?
NO
YES
NO
IS CORRESPONDING EQUIPMENT BUSY ?
YES
CLEAR PPU MESSAGE BUFFER
CLEAR PPU OUTPUT REGISTER
EXIT
NO
SEARCH EST FOR AN EQUIPMENT OF REQUESTED TYPE
IS THERE A PROPER TYPE FREE ?
YES
IS EQUIPMENT A DISK FILE ?
YES
NO
IS EQUIPMENT A DISK FILE ?
NO
YES
ASSIGN EQUIPMENT TO CONTROL POINT
SET EQUIPMENT ASSIGNMENT IN CONTROL POINT AREA
ENTER EQUIPMENT NUMBER IN PPU MESSAGE BUFFER
CLEAR PPU OUTPUT REGISTER
EXIT

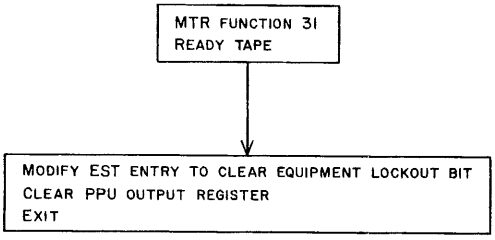

MTR FUNCTION 23
RELEASE EQUIPMENT
RELEASE EQUIPMENT ASSIGNMENT IN EST
CLEAR EQUIPMENT ASSIGNMENT IN CONTROL POINT AREA
CLEAR PPU OUTPUT REGISTER
EXIT

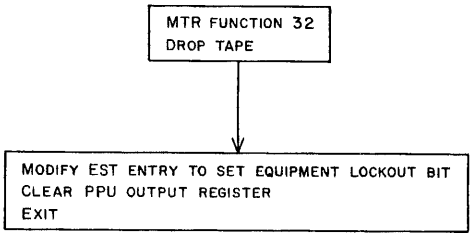

MTR FUNCTION 24
REQUEST PRIORITY
ENTER NEW PRIORITY IN CONTROL POINT AREA
RJ SEARCH FOR CP PRIORITY
CLEAR PPU OUTPUT REGISTER
EXIT

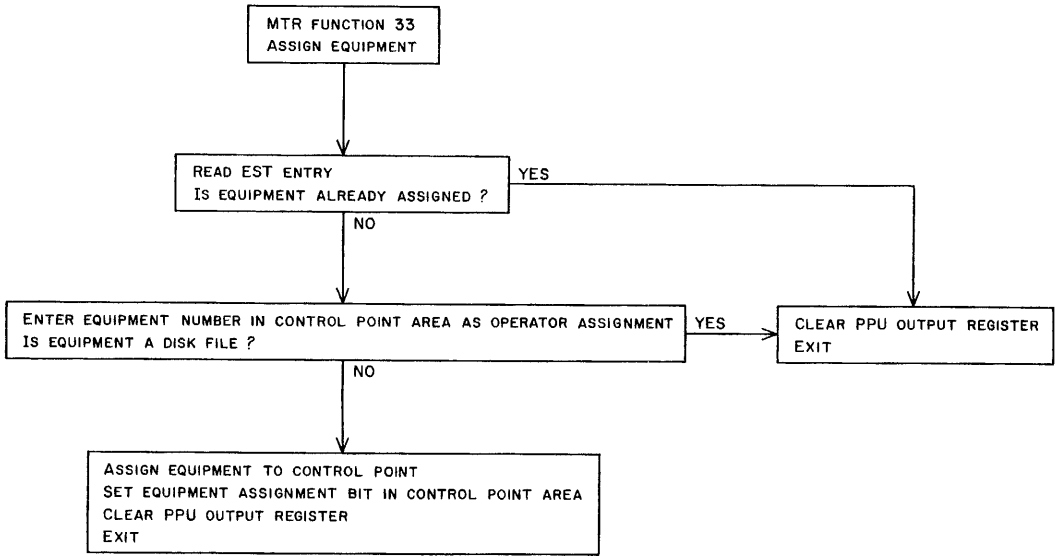

MTR FUNCTION 25
REQUEST EXIT MODE
IS CONTROL POINT IN CPU STACK ?
NO
CLEAR W FLAG AND X FLAG FOR CONTROL POINT
YES
EXCHANGE CURRENT CPU PROGRAM
PUSH UP CPU STACK
WAS REQUESTING CONTROL POINT EXCHANGED ?
YES
RJ SEARCH FOR CP PRIORITY
NO
SET W FLAG FOR EXCHANGED CONTROL POINT
RJ ADVANCE CLOCK
ENTER NEW EXIT MODE IN EXCHANGE PACKAGE
CLEAR PPU OUTPUT REGISTER
EXIT

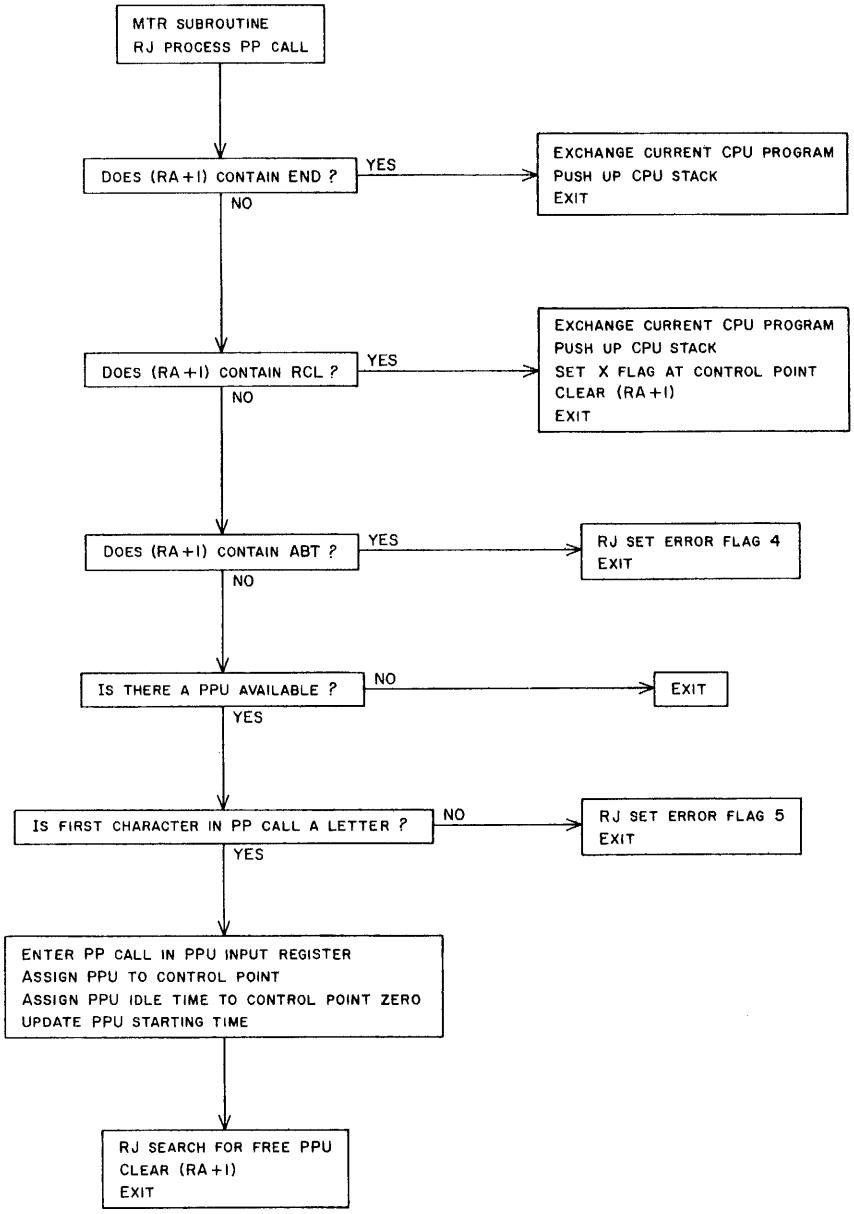

MTR FUNCTION 27
TOGGLE SIMULATOR STATUS
IS THERE A PPU AVAILABLE ?
NO
EXIT
YES
RESET EXCHANGE AREA FOR IDLE PROGRAM
IS SIMULATOR CURRENTLY OPERATING ?
YES
EXCHANGE SIMULATOR TO IDLE PROGRAM
CLEAR INPUT REGISTER FOR SIMULATOR PPU
WAIT FOR SIMULATOR TO FINISH
NO
EXCHANGE TO IDLE PROGRAM
ENTER SIMULATOR CALL IN PPU INPUT REGISTER
RJ SEARCH FOR FREE PPU
EXCHANGE CPU TO IDLE PROGRAM
MODIFY MONITOR PROGRAM TO TOGGLE SIMULATOR REFERENCES
CLEAR MONITOR FLAG
CLEAR PPU OUTPUT REGISTER
EXIT
MODIFY MONITOR PROGRAM TO TOGGLE ALL SIMULATOR REFERENCES
SET MONITOR FLAG
CLEAR PPU OUTPUT REGISTER
EXIT

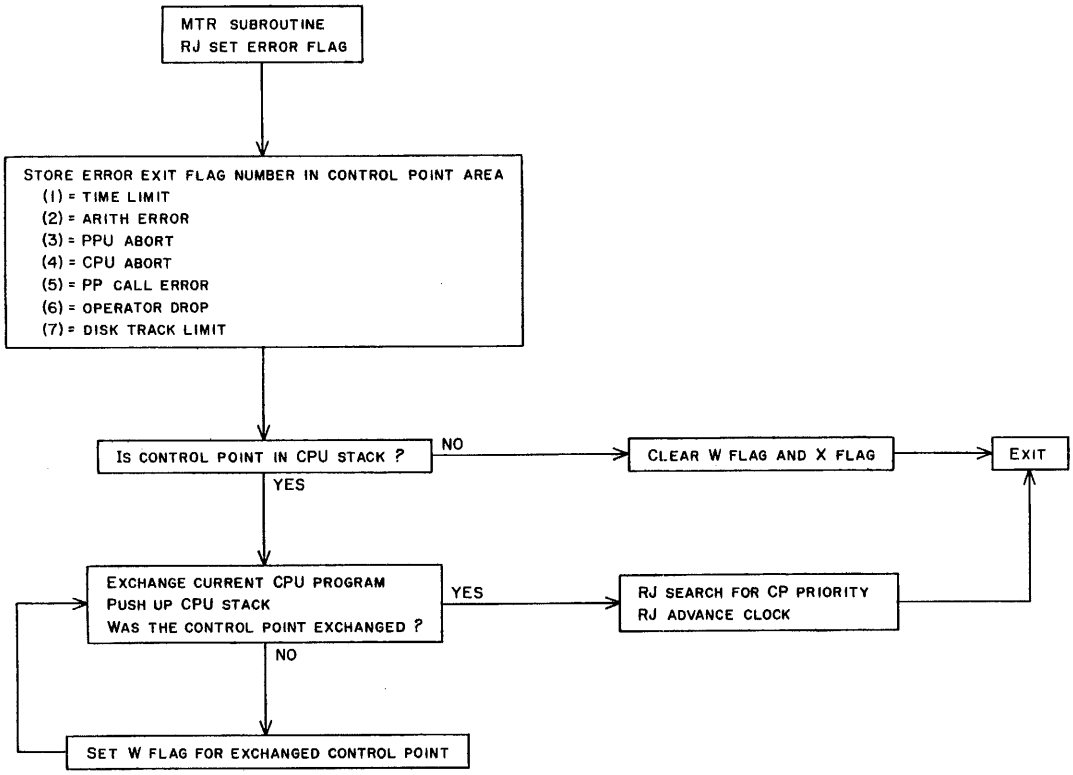

MTR FUNCTION 30
OPERATOR DROP
RJ SET ERROR FLAG 6
CLEAR PPU OUTPUT REGISTER
EXIT

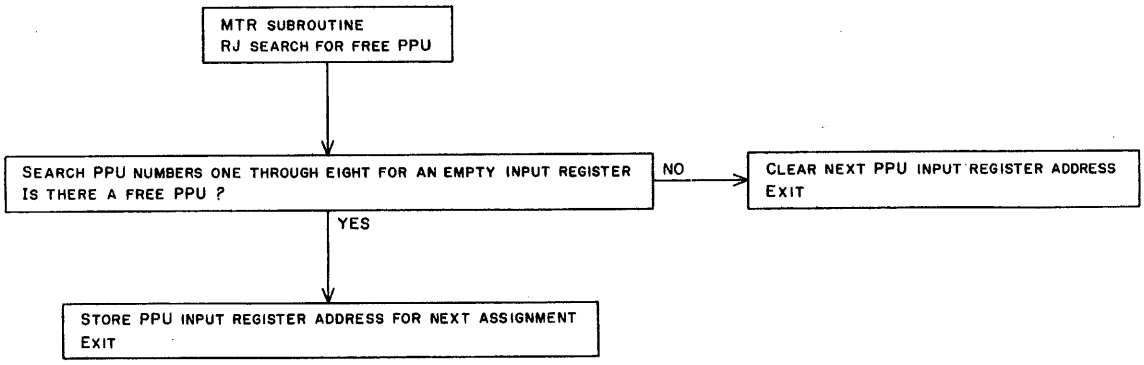

MTR FUNCTION 31
READY TAPE
MODIFY EST ENTRY TO CLEAR EQUIPMENT LOCKOUT BIT
CLEAR PPU OUTPUT REGISTER
EXIT

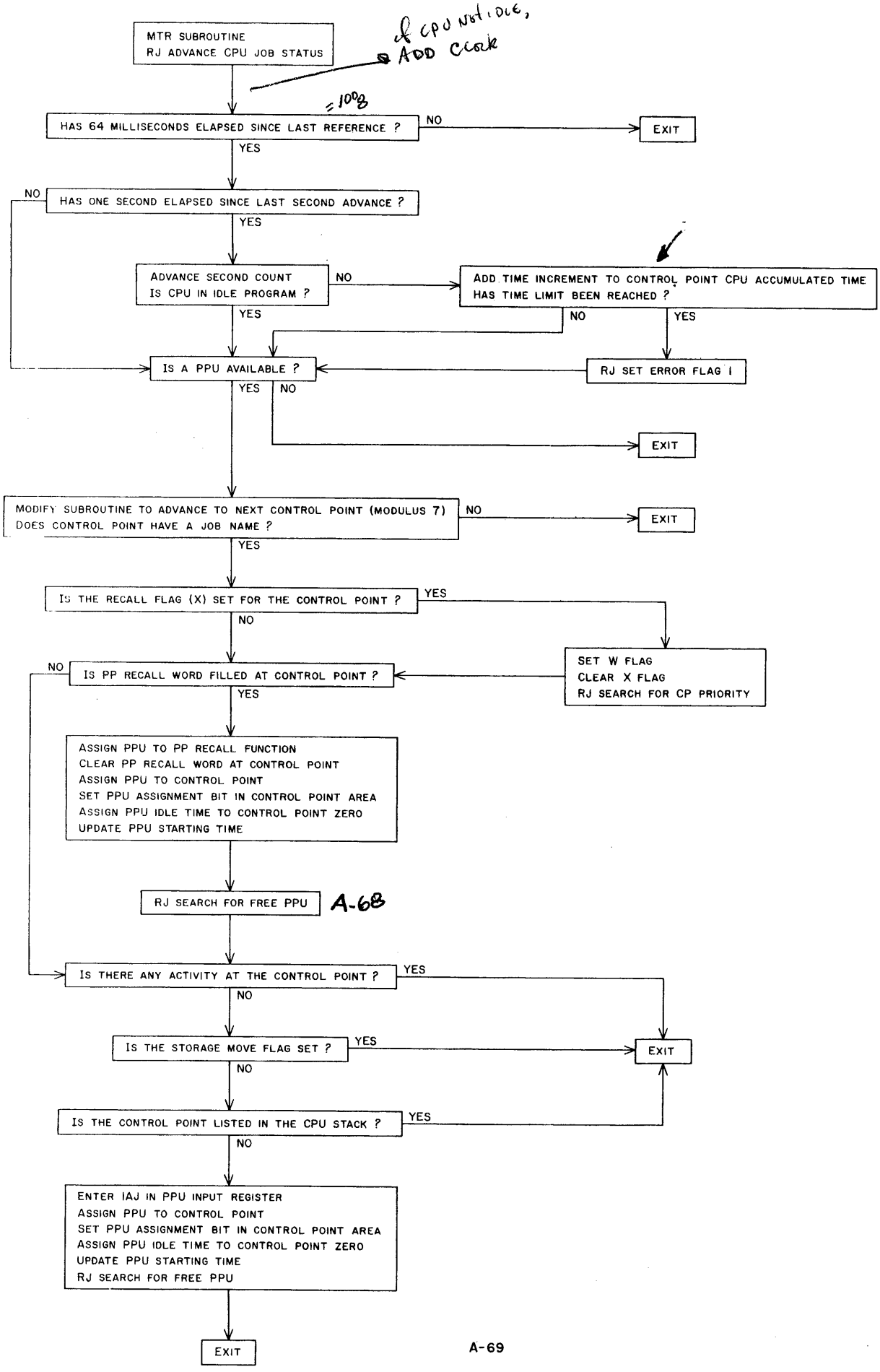

MTR FUNCTION 32
DROP TAPE
MODIFY EST ENTRY TO SET EQUIPMENT LOCKOUT BIT
CLEAR PPU OUTPUT REGISTER
EXIT

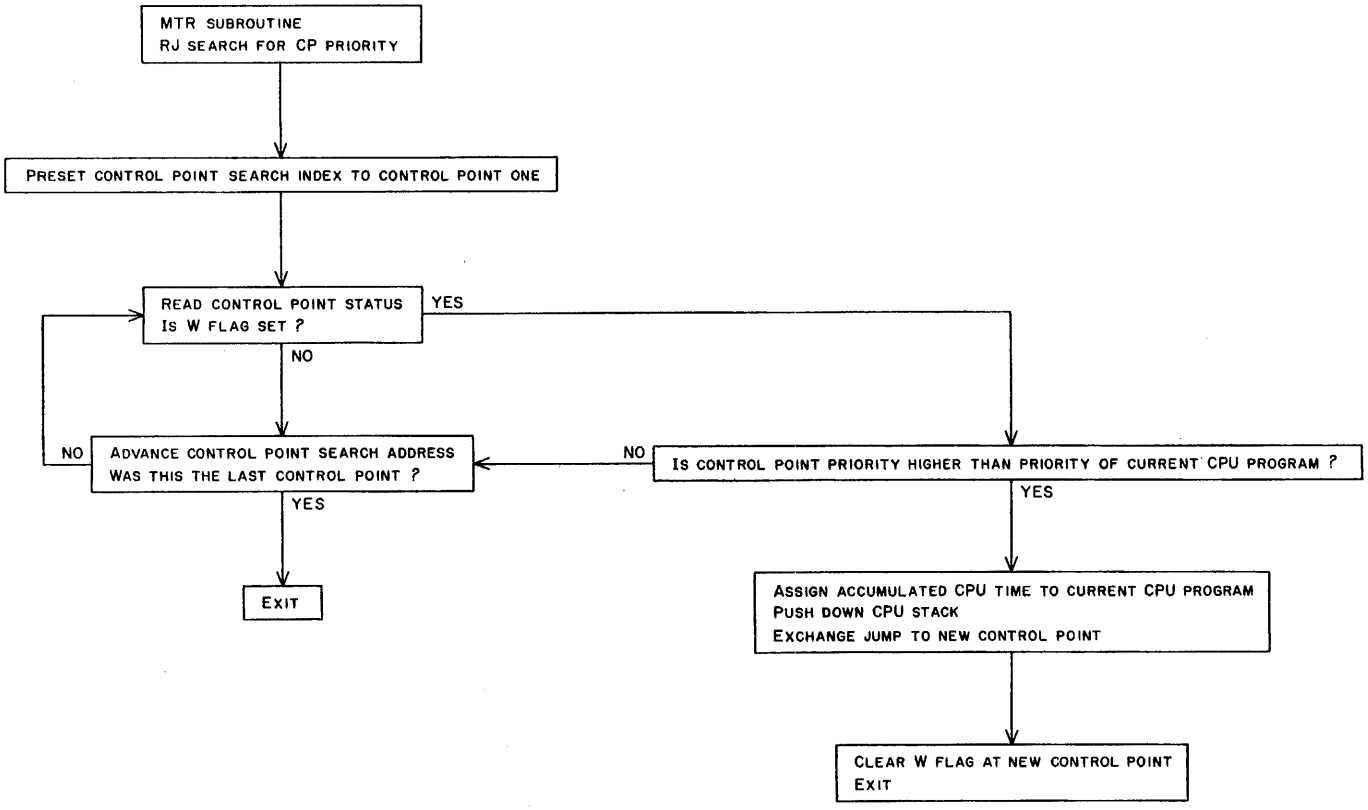

MTR FUNCTION 33
ASSIGN EQUIPMENT
READ EST ENTRY
IS EQUIPMENT ALREADY ASSIGNED ?
YES
NO
ENTER EQUIPMENT NUMBER IN CONTROL POINT AREA AS OPERATOR ASSIGNMENT
IS EQUIPMENT A DISK FILE ?
YES
CLEAR PPU OUTPUT REGISTER
EXIT
NO
ASSIGN EQUIPMENT TO CONTROL POINT
SET EQUIPMENT ASSIGNMENT BIT IN CONTROL POINT AREA
CLEAR PPU OUTPUT REGISTER
EXIT

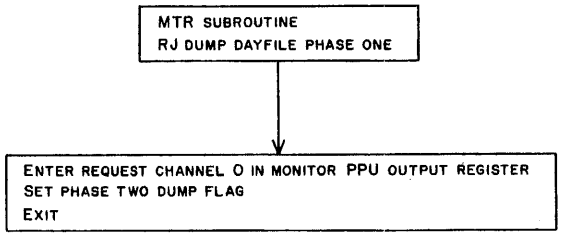

MTR SUBROUTINE
RJ PROCESS PP CALL
DOES (RA+1) CONTAIN END ?
YES
EXCHANGE CURRENT CPU PROGRAM
PUSH UP CPU STACK
EXIT
NO
DOES (RA+1) CONTAIN RCL ?
YES
EXCHANGE CURRENT CPU PROGRAM
PUSH UP CPU STACK
SET X FLAG AT CONTROL POINT
CLEAR (RA+1)
EXIT
NO
DOES (RA+1) CONTAIN ABT ?
YES
RJ SET ERROR FLAG 4
EXIT
NO
IS THERE A PPU AVAILABLE ?
NO
EXIT
YES
IS FIRST CHARACTER IN PP CALL A LETTER ?
NO
RJ SET ERROR FLAG 5
EXIT
YES
ENTER PP CALL IN PPU INPUT REGISTER
ASSIGN PPU TO CONTROL POINT
ASSIGN PPU IDLE TIME TO CONTROL POINT ZERO
UPDATE PPU STARTING TIME
RJ SEARCH FOR FREE PPU
CLEAR (RA+1)
EXIT

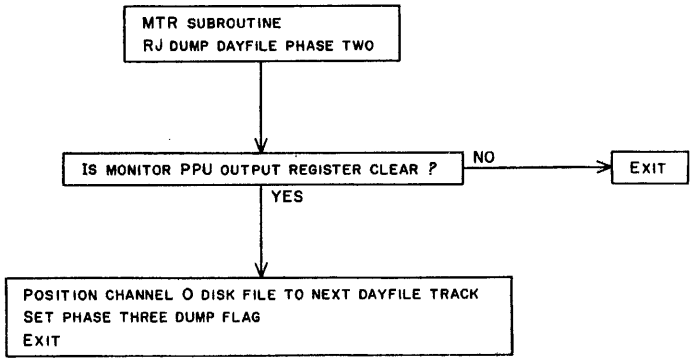

MTR SUBROUTINE
RJ SET ERROR FLAG
STORE ERROR EXIT FLAG NUMBER IN CONTROL POINT AREA
(1) = TIME LIMIT
(2) = ARITH ERROR
(3) = PPU ABORT
(4) = CPU ABORT
(5) = PP CALL ERROR
(6) = OPERATOR DROP
(7) = DISK TRACK LIMIT
IS CONTROL POINT IN CPU STACK ?
NO
CLEAR W FLAG AND X FLAG
EXIT
YES
EXCHANGE CURRENT CPU PROGRAM
PUSH UP CPU STACK
WAS THE CONTROL POINT EXCHANGED ?
YES
RJ SEARCH FOR CP PRIORITY
RJ ADVANCE CLOCK
NO
SET W FLAG FOR EXCHANGED CONTROL POINT

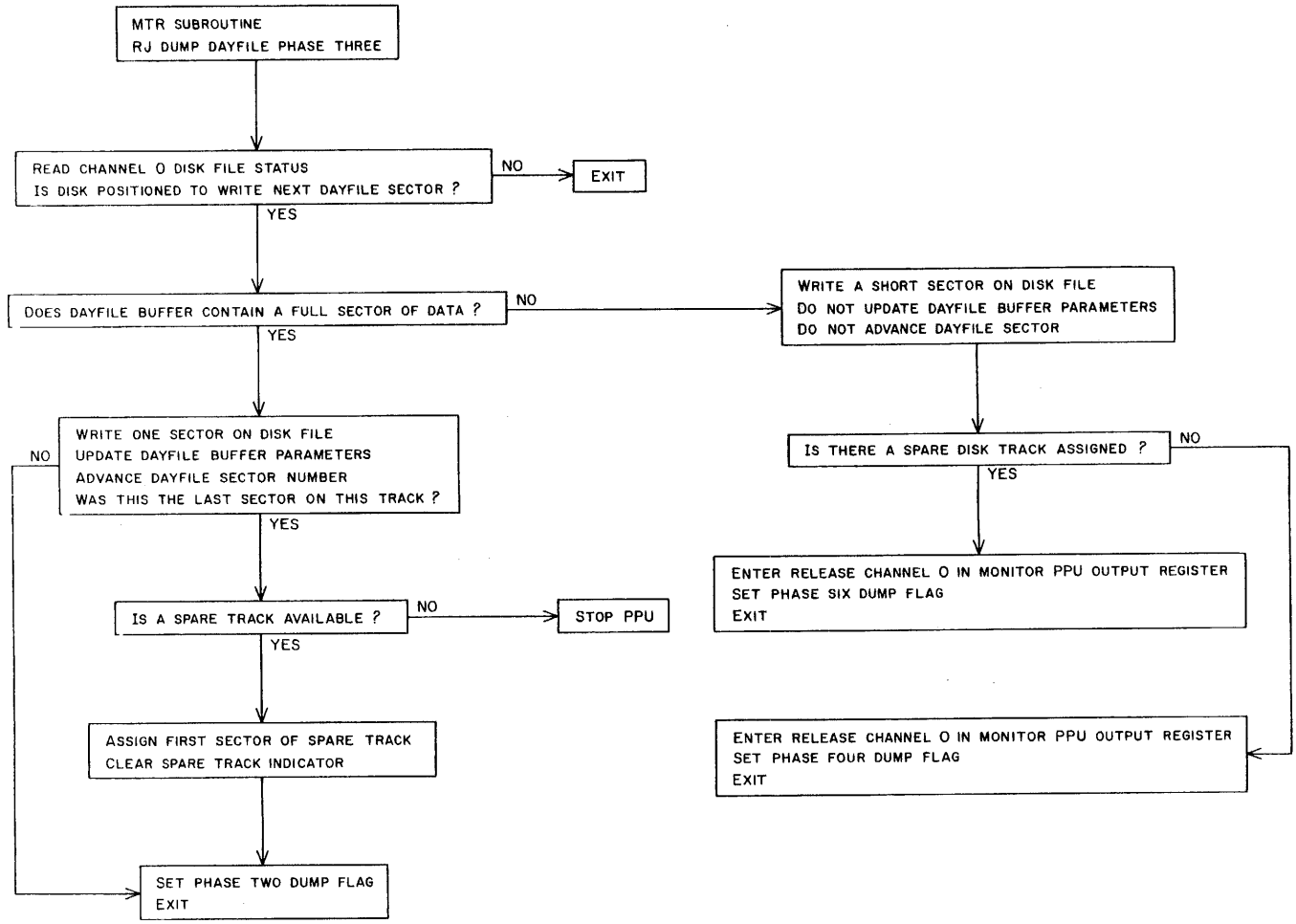

MTR SUBROUTINE
RJ SEARCH FOR FREE PPU
SEARCH PPU NUMBERS ONE THROUGH EIGHT FOR AN EMPTY INPUT REGISTER
IS THERE A FREE PPU ?
NO
CLEAR NEXT PPU INPUT REGISTER ADDRESS
EXIT
YES
STORE PPU INPUT REGISTER ADDRESS FOR NEXT ASSIGNMENT
EXIT

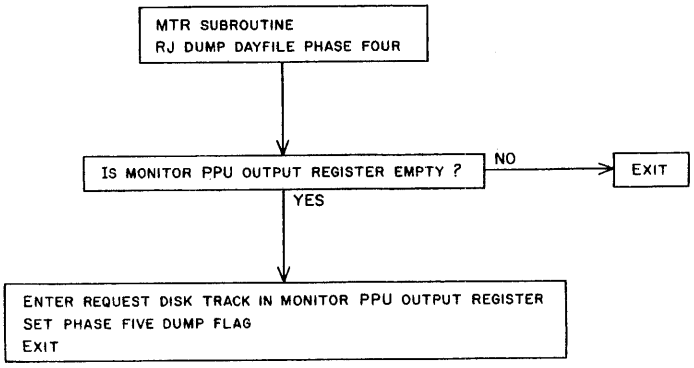

MTR SUBROUTINE
RJ ADVANCE CPU JOB STATUS
if CPU Not IDLE, ADD Clock
=1008
HAS 64 MILLISECONDS ELAPSED SINCE LAST REFERENCE ?
NO
EXIT
YES
NO
HAS ONE SECOND ELAPSED SINCE LAST SECOND ADVANCE ?
YES
ADVANCE SECOND COUNT
IS CPU IN IDLE PROGRAM ?
NO
ADD TIME INCREMENT TO CONTROL POINT CPU ACCUMULATED TIME
HAS TIME LIMIT BEEN REACHED ?
YES
NO
YES
IS A PPU AVAILABLE ?
RJ SET ERROR FLAG 1
YES
NO
EXIT
MODIFY SUBROUTINE TO ADVANCE TO NEXT CONTROL POINT (MODULUS 7)
DOES CONTROL POINT HAVE A JOB NAME ?
NO
EXIT
YES
IS THE RECALL FLAG (X) SET FOR THE CONTROL POINT ?
YES
NO
SET W FLAG
CLEAR X FLAG
RJ SEARCH FOR CP PRIORITY
NO
IS PP RECALL WORD FILLED AT CONTROL POINT ?
YES
ASSIGN PPU TO PP RECALL FUNCTION
CLEAR PP RECALL WORD AT CONTROL POINT
ASSIGN PPU TO CONTROL POINT
SET PPU ASSIGNMENT BIT IN CONTROL POINT AREA
ASSIGN PPU IDLE TIME TO CONTROL POINT ZERO
UPDATE PPU STARTING TIME
RJ SEARCH FOR FREE PPU
A-68
IS THERE ANY ACTIVITY AT THE CONTROL POINT ?
YES
NO
IS THE STORAGE MOVE FLAG SET ?
YES
EXIT
NO
IS THE CONTROL POINT LISTED IN THE CPU STACK ?
YES
NO
ENTER IAJ IN PPU INPUT REGISTER
ASSIGN PPU TO CONTROL POINT
SET PPU ASSIGNMENT BIT IN CONTROL POINT AREA
ASSIGN PPU IDLE TIME TO CONTROL POINT ZERO
UPDATE PPU STARTING TIME
RJ SEARCH FOR FREE PPU
EXIT

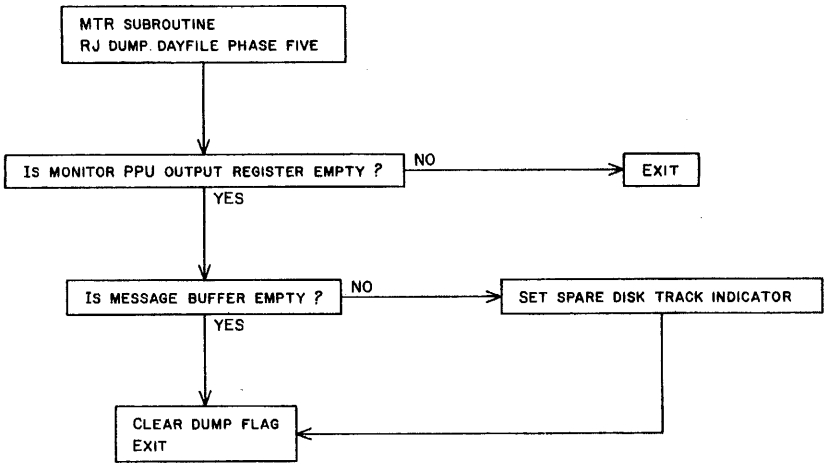

MTR SUBROUTINE
RJ SEARCH FOR CP PRIORITY
PRESET CONTROL POINT SEARCH INDEX TO CONTROL POINT ONE
READ CONTROL POINT STATUS
IS W FLAG SET ?
YES
NO
NO
ADVANCE CONTROL POINT SEARCH ADDRESS
WAS THIS THE LAST CONTROL POINT ?
YES
EXIT
NO
IS CONTROL POINT PRIORITY HIGHER THAN PRIORITY OF CURRENT CPU PROGRAM ?
YES
ASSIGN ACCUMULATED CPU TIME TO CURRENT CPU PROGRAM
PUSH DOWN CPU STACK
EXCHANGE JUMP TO NEW CONTROL POINT
CLEAR W FLAG AT NEW CONTROL POINT
EXIT

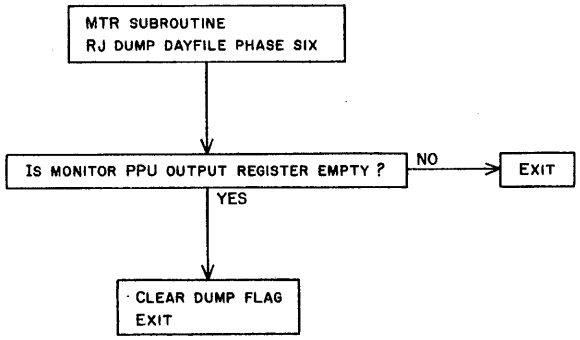

MTR SUBROUTINE
RJ DUMP DAYFILE PHASE ONE
ENTER REQUEST CHANNEL 0 IN MONITOR PPU OUTPUT REGISTER
SET PHASE TWO DUMP FLAG
EXIT


MTR SUBROUTINE
RJ DUMP DAYFILE PHASE TWO
IS MONITOR PPU OUTPUT REGISTER CLEAR ?
NO
EXIT
YES
POSITION CHANNEL 0 DISK FILE TO NEXT DAYFILE TRACK
SET PHASE THREE DUMP FLAG
EXIT


MTR SUBROUTINE
RJ DUMP DAYFILE PHASE THREE
READ CHANNEL 0 DISK FILE STATUS
IS DISK POSITIONED TO WRITE NEXT DAYFILE SECTOR ?
NO
EXIT
YES
DOES DAYFILE BUFFER CONTAIN A FULL SECTOR OF DATA ?
NO
YES
WRITE A SHORT SECTOR ON DISK FILE
DO NOT UPDATE DAYFILE BUFFER PARAMETERS
DO NOT ADVANCE DAYFILE SECTOR
WRITE ONE SECTOR ON DISK FILE
UPDATE DAYFILE BUFFER PARAMETERS
ADVANCE DAYFILE SECTOR NUMBER
WAS THIS THE LAST SECTOR ON THIS TRACK ?
NO
YES
IS THERE A SPARE DISK TRACK ASSIGNED ?
NO
YES
IS A SPARE TRACK AVAILABLE ?
NO
STOP PPU
YES
ENTER RELEASE CHANNEL 0 IN MONITOR PPU OUTPUT REGISTER
SET PHASE SIX DUMP FLAG
EXIT
ASSIGN FIRST SECTOR OF SPARE TRACK
CLEAR SPARE TRACK INDICATOR
ENTER RELEASE CHANNEL 0 IN MONITOR PPU OUTPUT REGISTER
SET PHASE FOUR DUMP FLAG
EXIT
SET PHASE TWO DUMP FLAG
EXIT

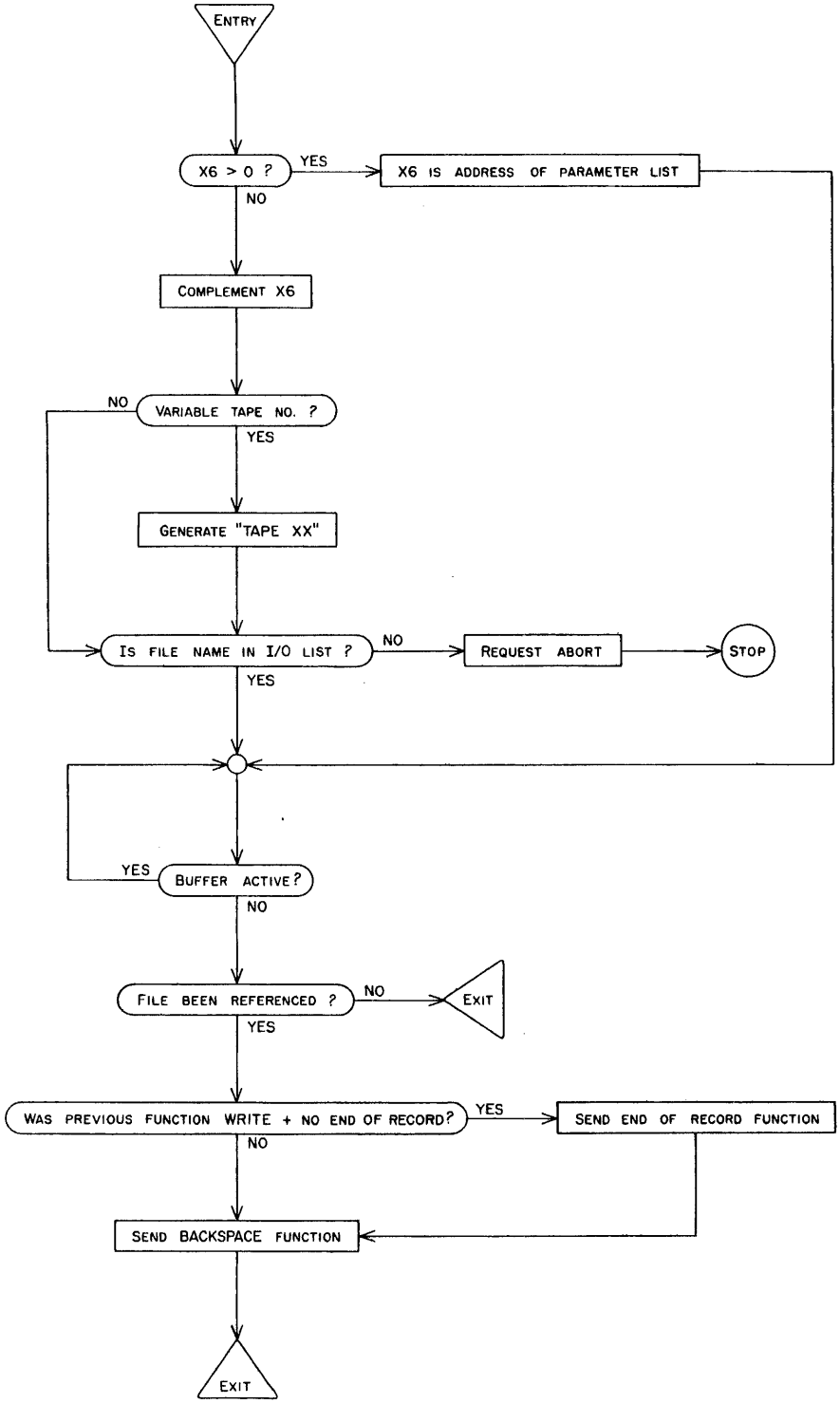

MTR SUBROUTINE
RJ DUMP DAYFILE PHASE FOUR
IS MONITOR PPU OUTPUT REGISTER EMPTY ?
NO
EXIT
YES
ENTER REQUEST DISK TRACK IN MONITOR PPU OUTPUT REGISTER
SET PHASE FIVE DUMP FLAG
EXIT

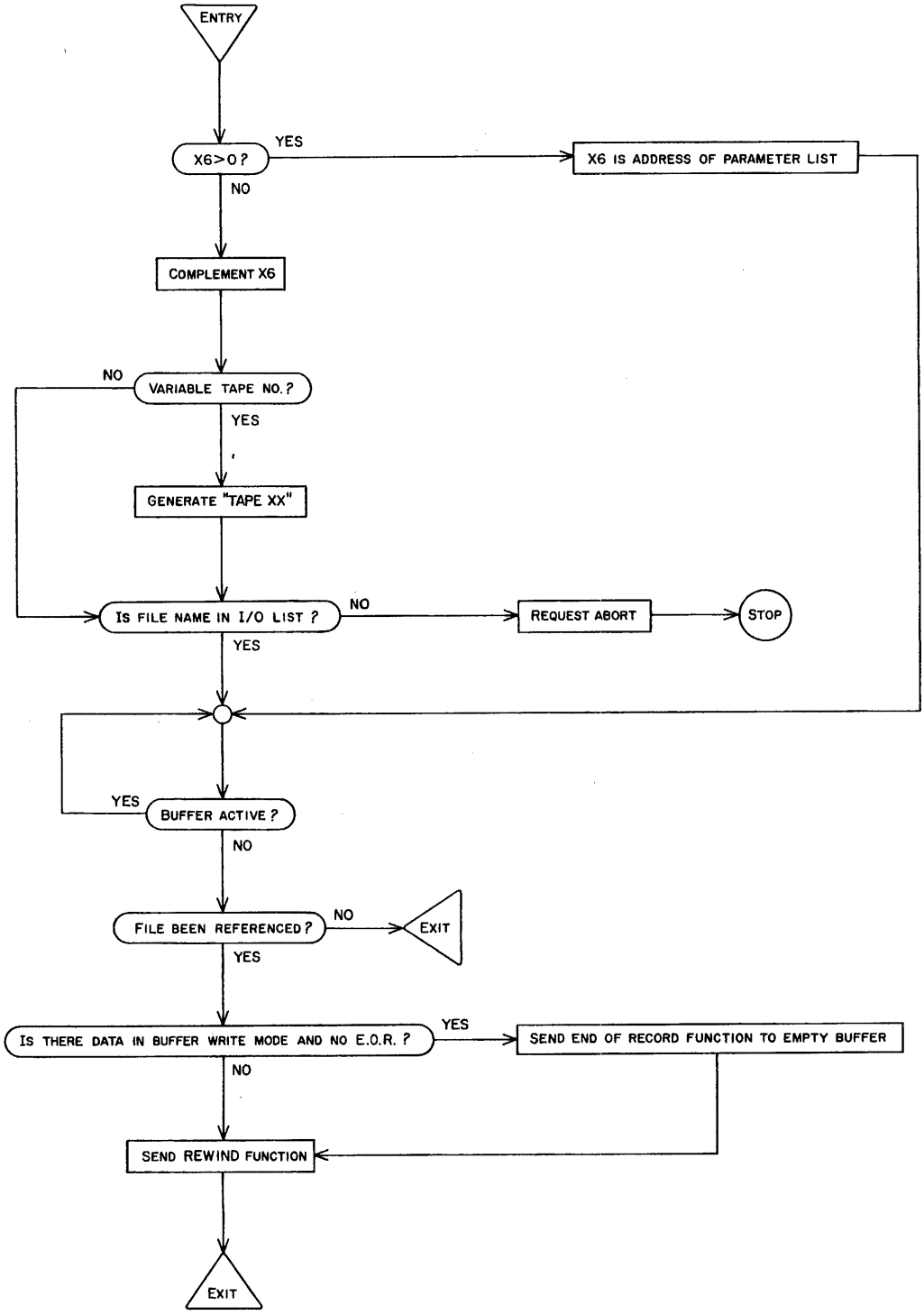

MTR SUBROUTINE
RJ DUMP. DAYFILE PHASE FIVE
IS MONITOR PPU OUTPUT REGISTER EMPTY ?
NO
EXIT
YES
IS MESSAGE BUFFER EMPTY ?
NO
SET SPARE DISK TRACK INDICATOR
YES
CLEAR DUMP FLAG
EXIT

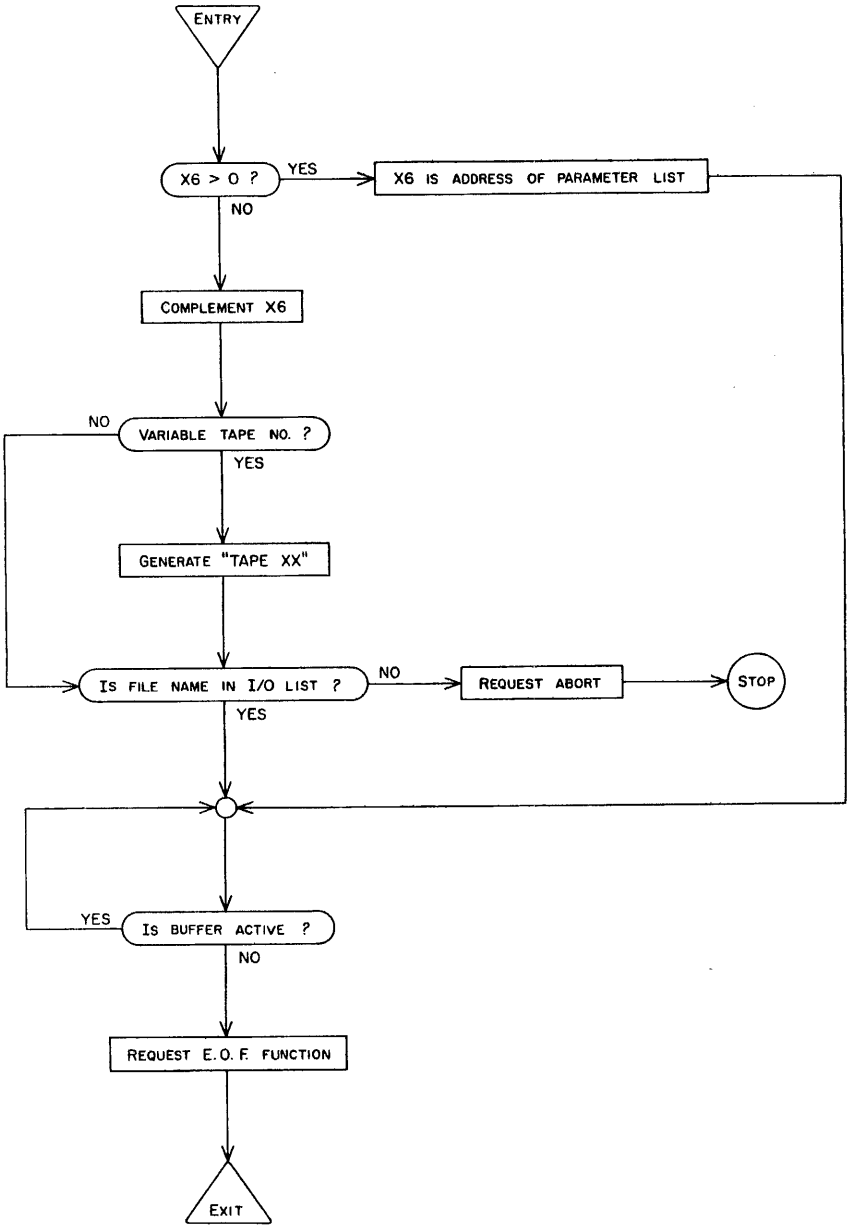

MTR SUBROUTINE
RJ DUMP DAYFILE PHASE SIX
IS MONITOR PPU OUTPUT REGISTER EMPTY ?
NO
EXIT
YES
CLEAR DUMP FLAG
EXIT

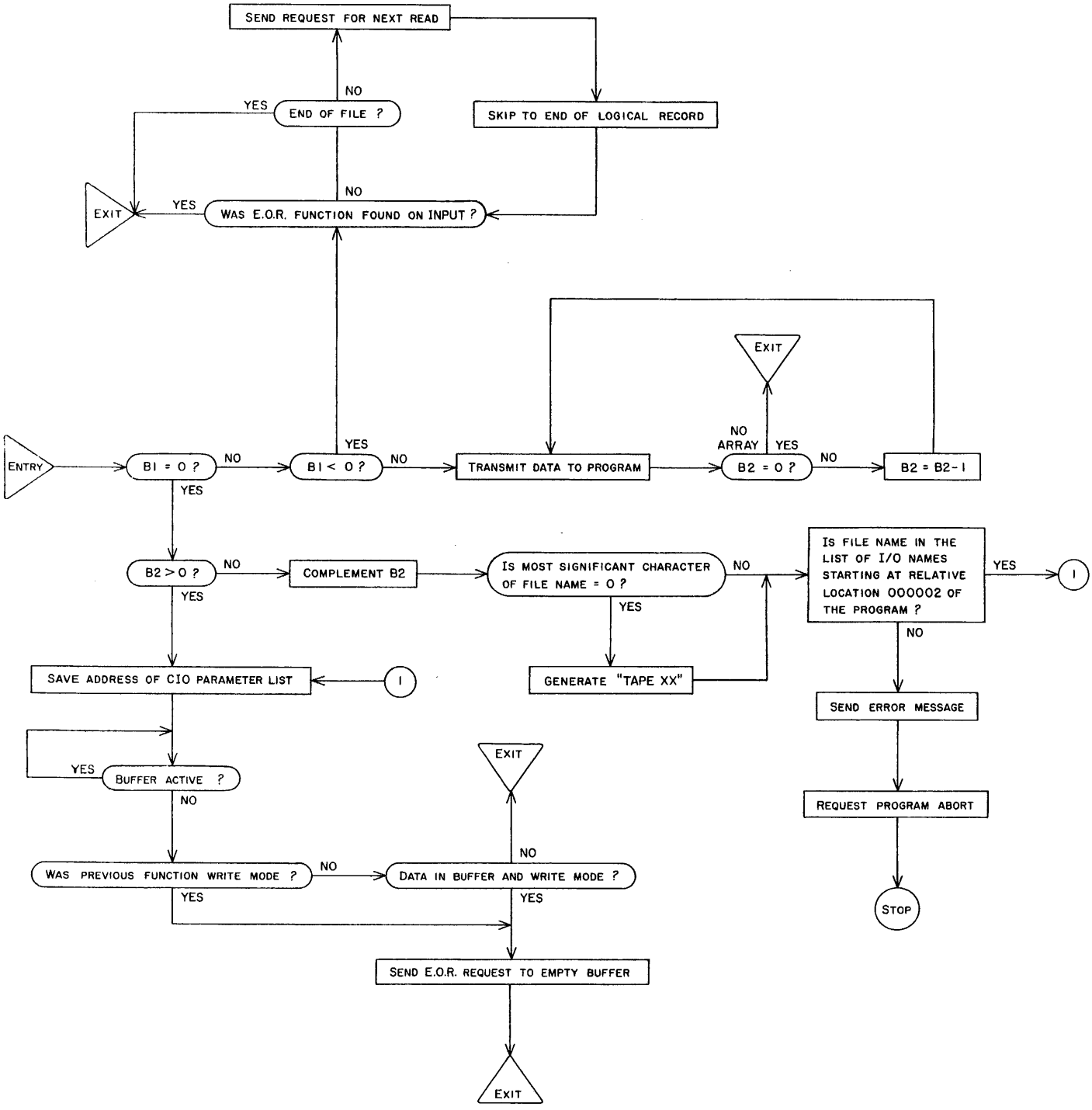
APPENDIX B

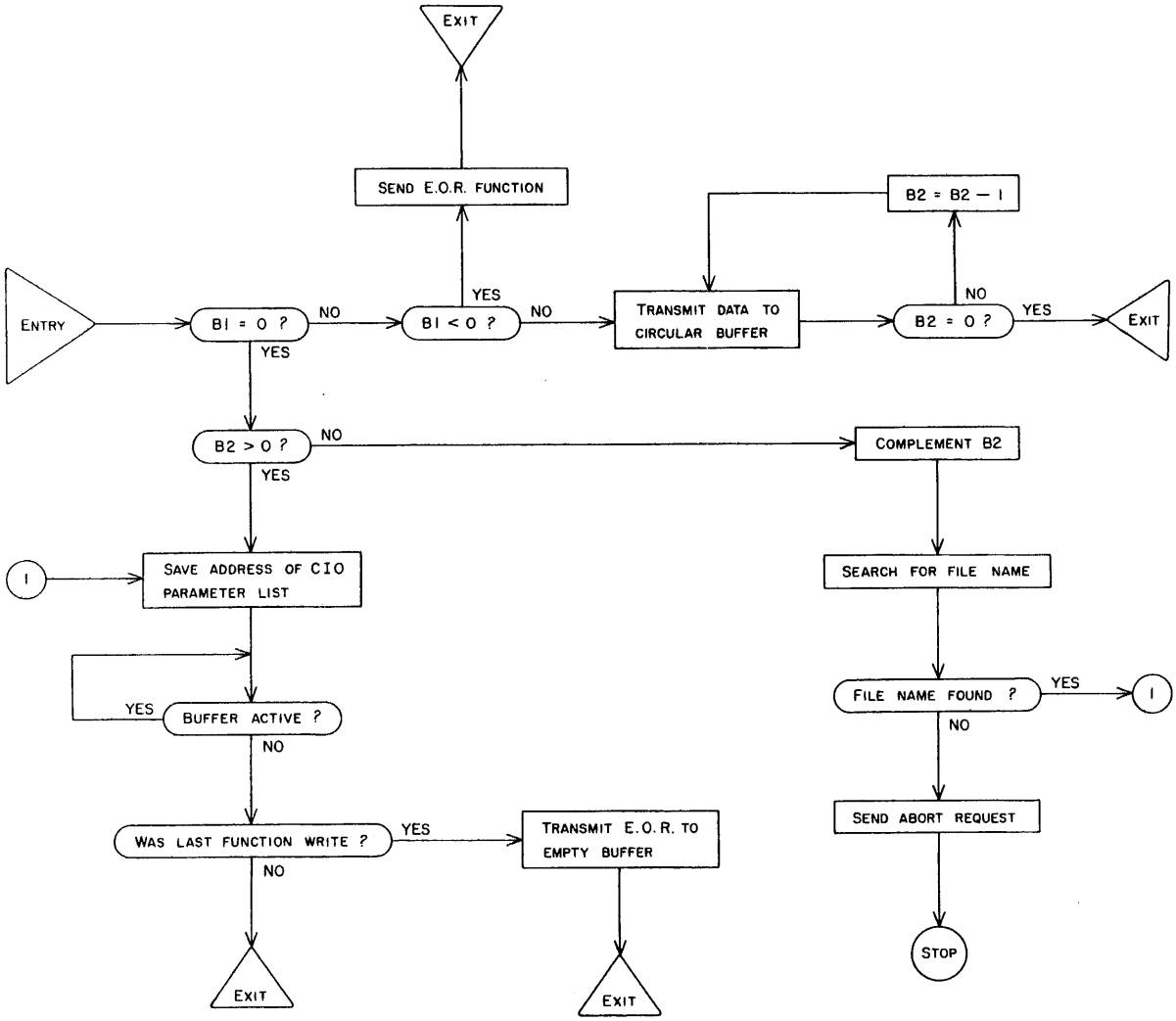
I/O SUBROUTINE FLOW CHARTS


ENTRY
X6 > 0 ?
YES
X6 IS ADDRESS OF PARAMETER LIST
NO
COMPLEMENT X6
NO
VARIABLE TAPE NO. ?
YES
GENERATE "TAPE XX"
IS FILE NAME IN I/O LIST ?
NO
REQUEST ABORT
STOP
YES
YES
BUFFER ACTIVE?
NO
FILE BEEN REFERENCED ?
NO
EXIT
YES
WAS PREVIOUS FUNCTION WRITE + NO END OF RECORD?
YES
SEND END OF RECORD FUNCTION
NO
SEND BACKSPACE FUNCTION
EXIT


ENTRY
X6>0 ?
YES
X6 IS ADDRESS OF PARAMETER LIST
NO
COMPLEMENT X6
NO
VARIABLE TAPE NO. ?
YES
GENERATE "TAPE XX"
IS FILE NAME IN I/O LIST ?
NO
REQUEST ABORT
STOP
YES
YES
BUFFER ACTIVE ?
NO
FILE BEEN REFERENCED ?
NO
EXIT
YES
IS THERE DATA IN BUFFER WRITE MODE AND NO E.O.R. ?
YES
SEND END OF RECORD FUNCTION TO EMPTY BUFFER
NO
SEND REWIND FUNCTION
EXIT

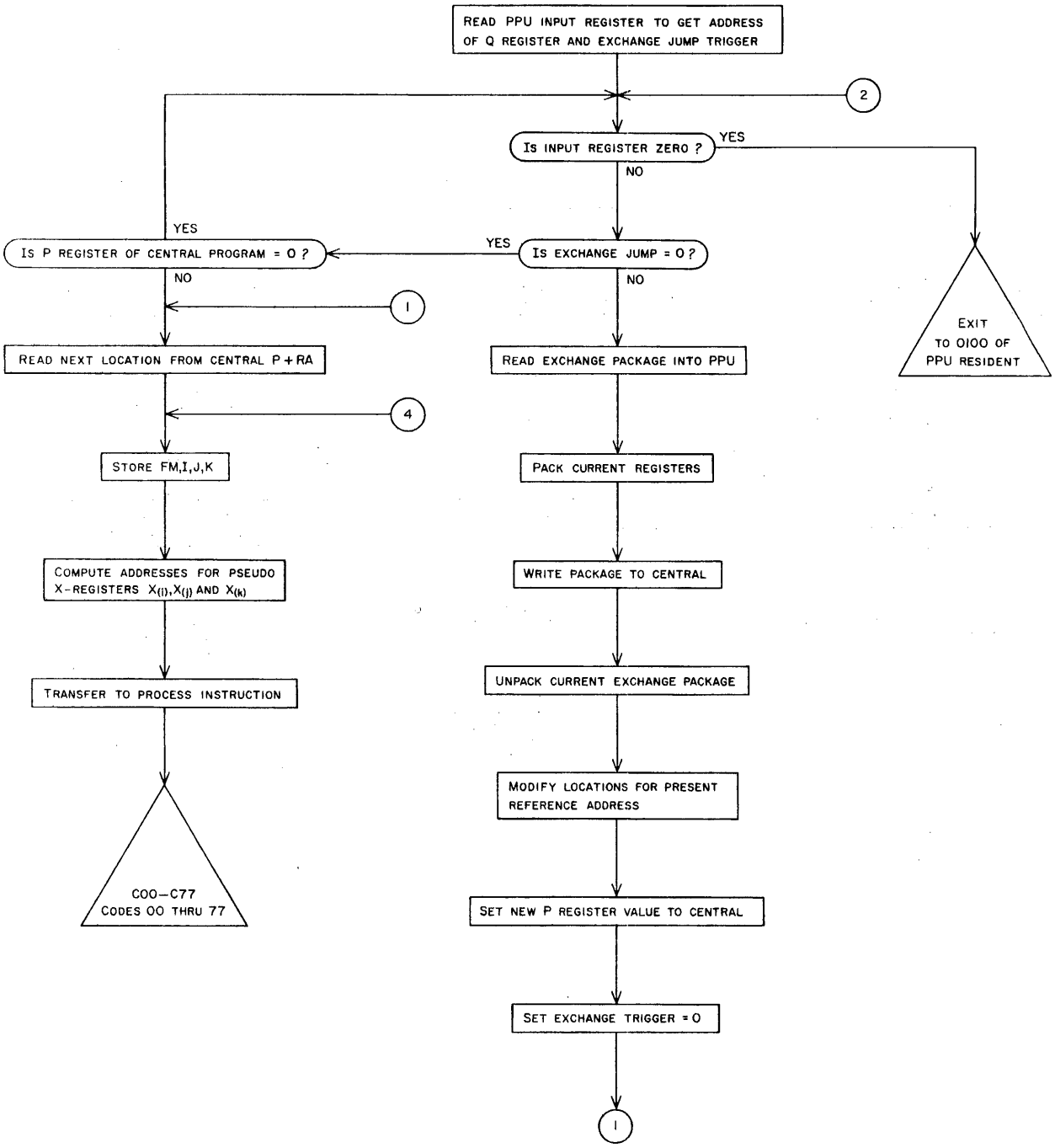

ENTRY
X6 > 0 ?
YES
X6 IS ADDRESS OF PARAMETER LIST
NO
COMPLEMENT X6
NO
VARIABLE TAPE NO. ?
YES
GENERATE "TAPE XX"
IS FILE NAME IN I/O LIST ?
NO
REQUEST ABORT
STOP
YES
YES
IS BUFFER ACTIVE ?
NO
REQUEST E.O.F. FUNCTION
EXIT

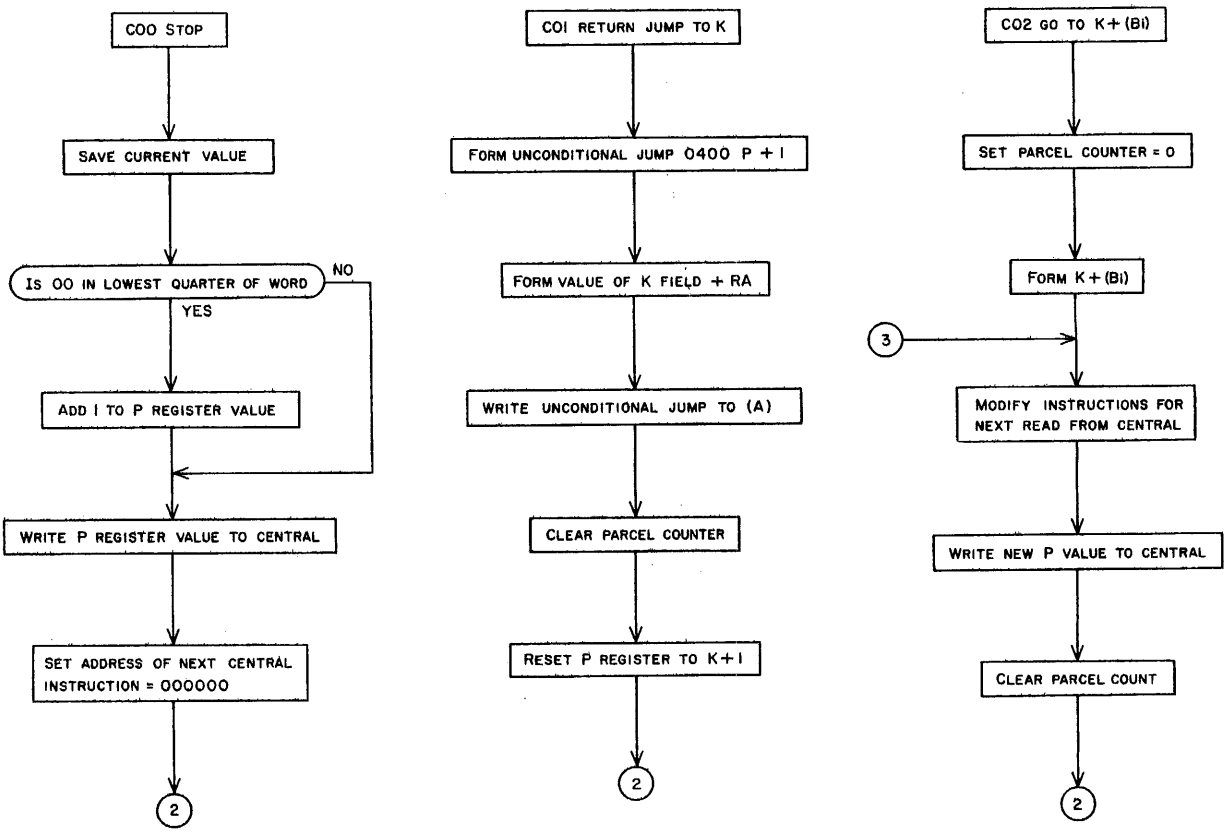

SEND REQUEST FOR NEXT READ
NO
YES
END OF FILE ?
SKIP TO END OF LOGICAL RECORD
NO
EXIT
YES
WAS E.O.R. FUNCTION FOUND ON INPUT ?
EXIT
NO
ARRAY
YES
ENTRY
B1 = 0 ?
NO
YES
B1 < 0 ?
NO
TRANSMIT DATA TO PROGRAM
B2 = 0 ?
NO
B2 = B2-1
YES
B2 > 0 ?
NO
COMPLEMENT B2
IS MOST SIGNIFICANT CHARACTER OF FILE NAME = 0 ?
NO
YES
IS FILE NAME IN THE LIST OF I/O NAMES STARTING AT RELATIVE LOCATION 000002 OF THE PROGRAM ?
YES
1
NO
YES
SAVE ADDRESS OF CIO PARAMETER LIST
1
GENERATE "TAPE XX"
SEND ERROR MESSAGE
YES
BUFFER ACTIVE ?
NO
EXIT
REQUEST PROGRAM ABORT
NO
WAS PREVIOUS FUNCTION WRITE MODE ?
NO
DATA IN BUFFER AND WRITE MODE ?
YES
YES
STOP
SEND E.O.R. REQUEST TO EMPTY BUFFER
EXIT

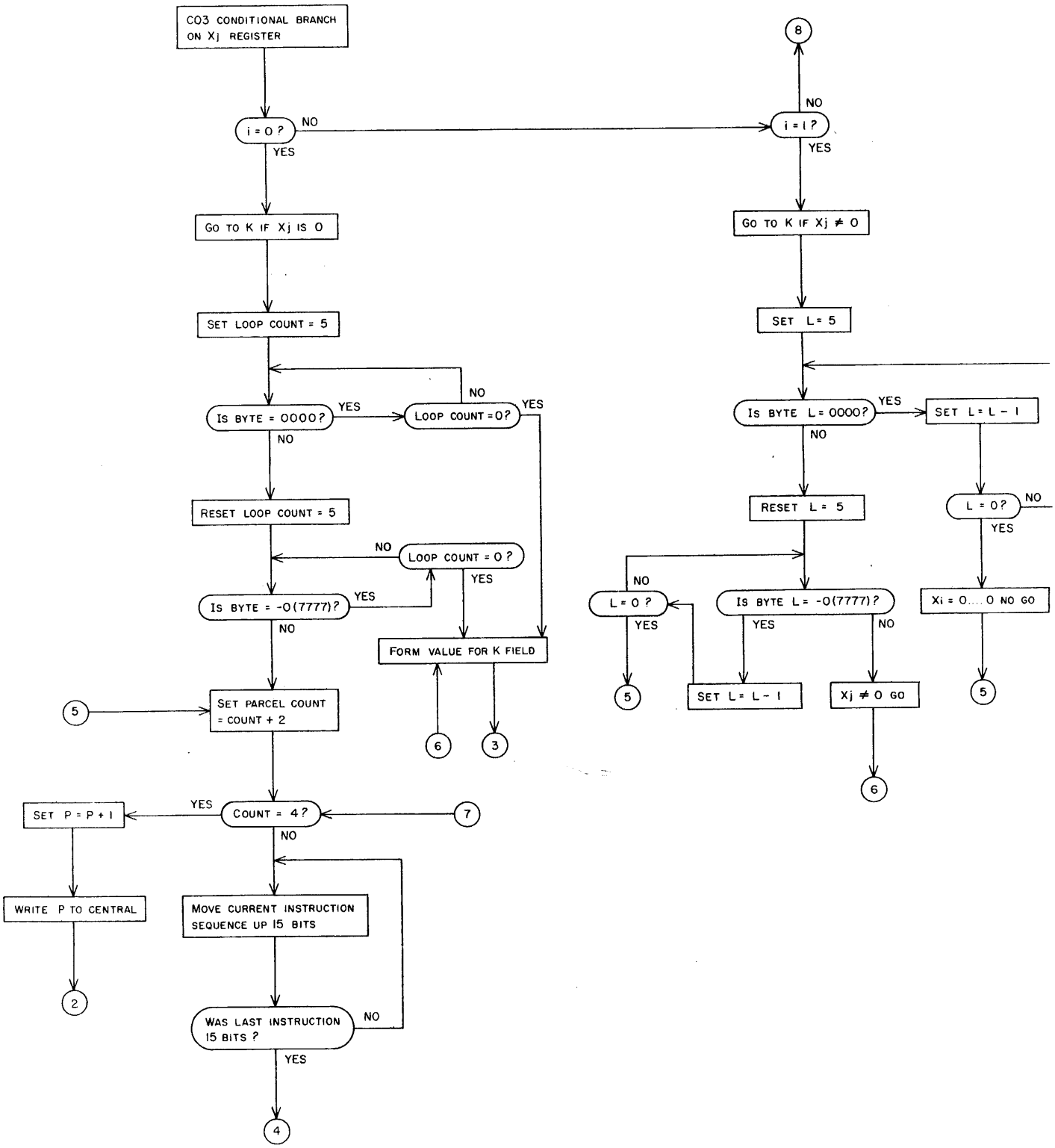

EXIT
SEND E.O.R. FUNCTION
B2 = B2 − 1
ENTRY
B1 = 0 ?
NO
B1 < 0 ?
NO
TRANSMIT DATA TO CIRCULAR BUFFER
B2 = 0 ?
NO
YES
EXIT
YES
YES
B2 > 0 ?
NO
COMPLEMENT B2
YES
1
SAVE ADDRESS OF CIO PARAMETER LIST
SEARCH FOR FILE NAME
YES
BUFFER ACTIVE ?
FILE NAME FOUND ?
YES
1
NO
NO
WAS LAST FUNCTION WRITE ?
YES
TRANSMIT E.O.R. TO EMPTY BUFFER
SEND ABORT REQUEST
NO
STOP
EXIT
EXIT

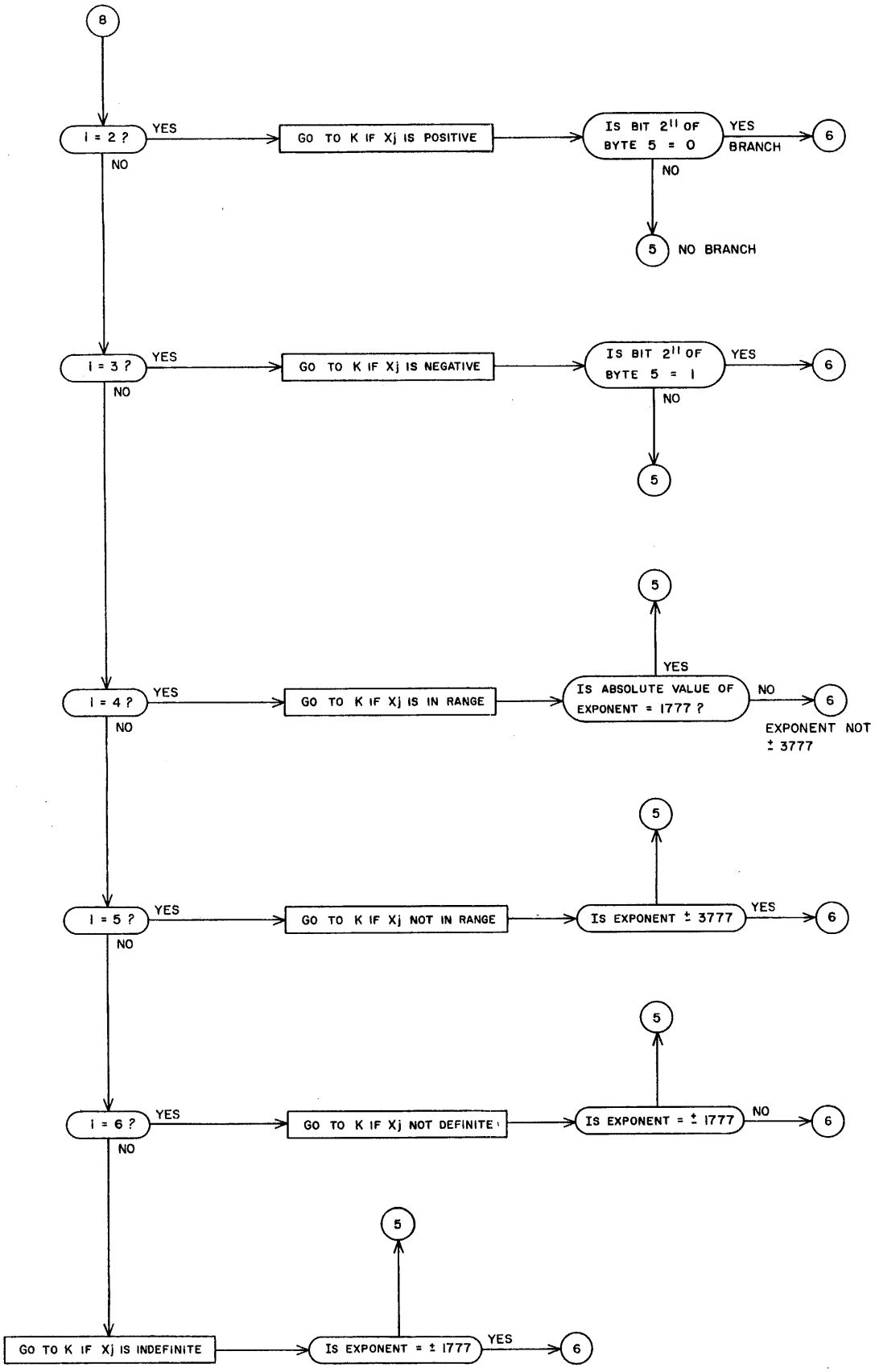
APPENDIX C

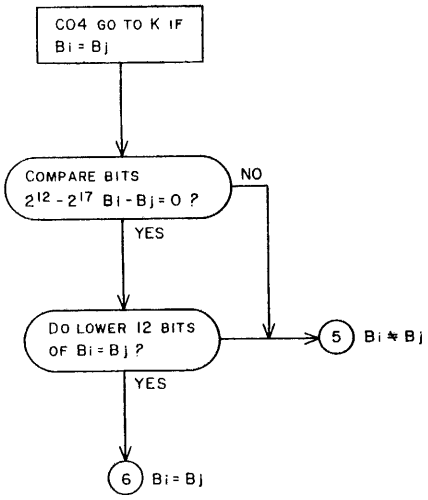
SIM FLOW CHARTS

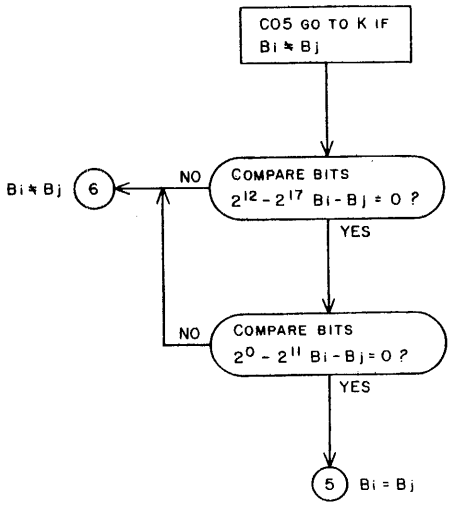

READ PPU INPUT REGISTER TO GET ADDRESS OF Q REGISTER AND EXCHANGE JUMP TRIGGER
2
IS INPUT REGISTER ZERO ?
YES
NO
EXIT TO 0100 OF PPU RESIDENT
IS EXCHANGE JUMP = 0 ?
YES
NO
IS P REGISTER OF CENTRAL PROGRAM = 0 ?
YES
NO
1
READ NEXT LOCATION FROM CENTRAL P + RA
READ EXCHANGE PACKAGE INTO PPU
4
STORE FM,I,J,K
PACK CURRENT REGISTERS
COMPUTE ADDRESSES FOR PSEUDO X-REGISTERS X(i),X(j) AND X(k)
WRITE PACKAGE TO CENTRAL
TRANSFER TO PROCESS INSTRUCTION
UNPACK CURRENT EXCHANGE PACKAGE
MODIFY LOCATIONS FOR PRESENT REFERENCE ADDRESS
C00–C77
CODES 00 THRU 77
SET NEW P REGISTER VALUE TO CENTRAL
SET EXCHANGE TRIGGER = 0
1

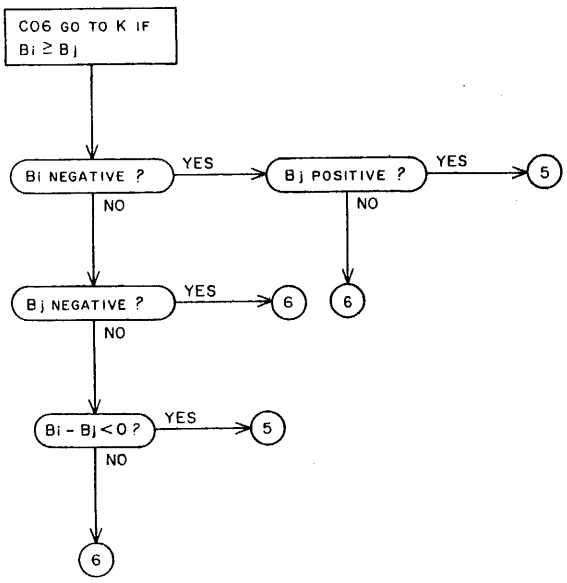

COO STOP
SAVE CURRENT VALUE
IS OO IN LOWEST QUARTER OF WORD
NO
YES
ADD I TO P REGISTER VALUE
WRITE P REGISTER VALUE TO CENTRAL
SET ADDRESS OF NEXT CENTRAL INSTRUCTION = 000000
2
COI RETURN JUMP TO K
FORM UNCONDITIONAL JUMP 0400 P + I
FORM VALUE OF K FIELD + RA
WRITE UNCONDITIONAL JUMP TO (A)
CLEAR PARCEL COUNTER
RESET P REGISTER TO K+I
2
CO2 GO TO K+ (Bi)
SET PARCEL COUNTER = 0
FORM K+ (Bi)
3
MODIFY INSTRUCTIONS FOR NEXT READ FROM CENTRAL
WRITE NEW P VALUE TO CENTRAL
CLEAR PARCEL COUNT
2

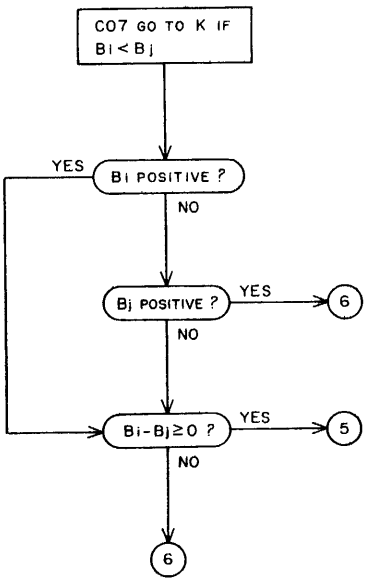

CO3 CONDITIONAL BRANCH ON Xj REGISTER
i = 0 ?
NO
YES
i = 1 ?
NO
YES
8
GO TO K IF Xj IS 0
GO TO K IF Xj ≠ 0
SET LOOP COUNT = 5
SET L = 5
IS BYTE = 0000?
YES
NO
LOOP COUNT = 0?
YES
NO
IS BYTE L = 0000?
YES
NO
SET L = L - 1
RESET LOOP COUNT = 5
RESET L = 5
L = 0?
NO
YES
LOOP COUNT = 0 ?
NO
YES
IS BYTE = -0(7777)?
YES
NO
L = 0 ?
NO
YES
IS BYTE L = -0(7777)?
YES
NO
Xi = 0.... 0 NO GO
FORM VALUE FOR K FIELD
5
SET PARCEL COUNT = COUNT + 2
6
3
5
SET L = L - 1
Xj ≠ 0 GO
5
6
SET P = P + 1
YES
COUNT = 4?
7
NO
WRITE P TO CENTRAL
MOVE CURRENT INSTRUCTION SEQUENCE UP 15 BITS
2
WAS LAST INSTRUCTION 15 BITS ?
NO
YES
4

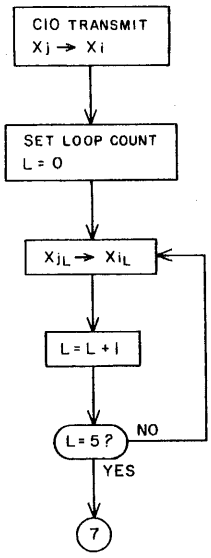

8
i = 2 ?
YES
NO
GO TO K IF Xj IS POSITIVE
IS BIT 2^11 OF BYTE 5 = 0
YES
BRANCH
6
NO
5
NO BRANCH
i = 3 ?
YES
NO
GO TO K IF Xj IS NEGATIVE
IS BIT 2^11 OF BYTE 5 = 1
YES
6
NO
5
i = 4 ?
YES
NO
GO TO K IF Xj IS IN RANGE
5
YES
IS ABSOLUTE VALUE OF EXPONENT = 1777 ?
NO
6
EXPONENT NOT ± 3777
i = 5 ?
YES
NO
GO TO K IF Xj NOT IN RANGE
5
IS EXPONENT ± 3777
YES
6
i = 6 ?
YES
NO
GO TO K IF Xj NOT DEFINITE
5
IS EXPONENT = ± 1777
NO
6
GO TO K IF Xj IS INDEFINITE
5
IS EXPONENT = ± 1777
YES
6

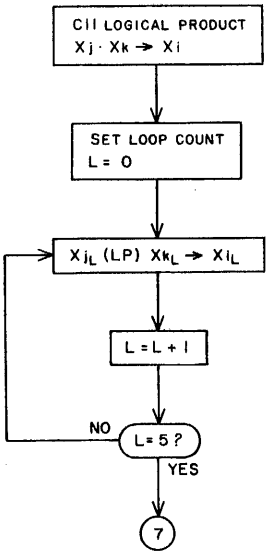

CO4 GO TO K IF
Bi = Bj
COMPARE BITS
$2^{12} - 2^{17}$ Bi - Bj = 0 ?
NO
YES
DO LOWER 12 BITS
OF Bi = Bj ?
5 Bi ≠ Bj
YES
6 Bi = Bj

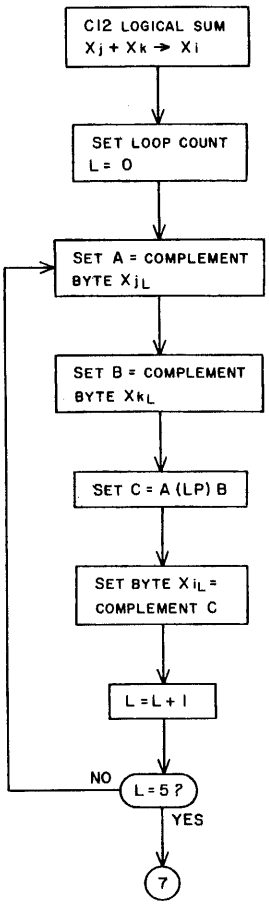

CO5 GO TO K IF
Bi ≠ Bj
Bi ≠ Bj 6
NO
COMPARE BITS
$2^{12} - 2^{17}$ Bi - Bj = 0 ?
YES
NO
COMPARE BITS
$2^{0} - 2^{11}$ Bi - Bj = 0 ?
YES
5 Bi = Bj

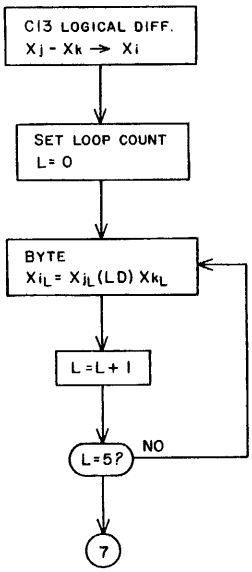

CO6 GO TO K IF
Bi ≥ Bj
Bi NEGATIVE ?
YES
Bj POSITIVE ?
YES
5
NO
NO
Bj NEGATIVE ?
YES
6
6
NO
Bi - Bj < 0 ?
YES
5
NO
6

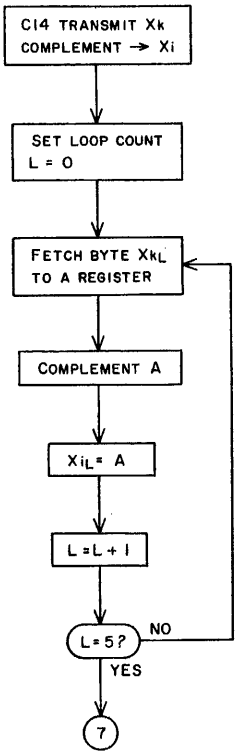

CO7 GO TO K IF
Bi < Bj
YES
Bi POSITIVE ?
NO
Bj POSITIVE ?
YES
6
NO
Bi-Bj ≥ 0 ?
YES
5
NO
6

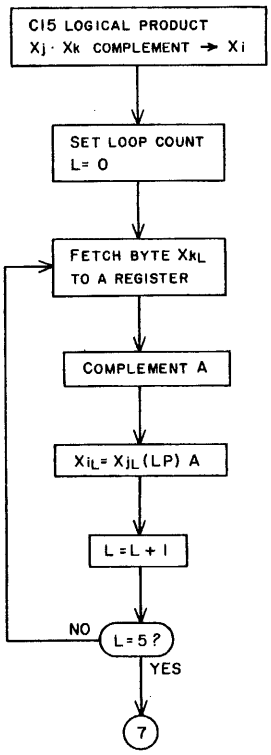

C10 TRANSMIT
Xj → Xi
SET LOOP COUNT
L = 0
XjL → XiL
L = L + 1
L = 5 ?
NO
YES
7

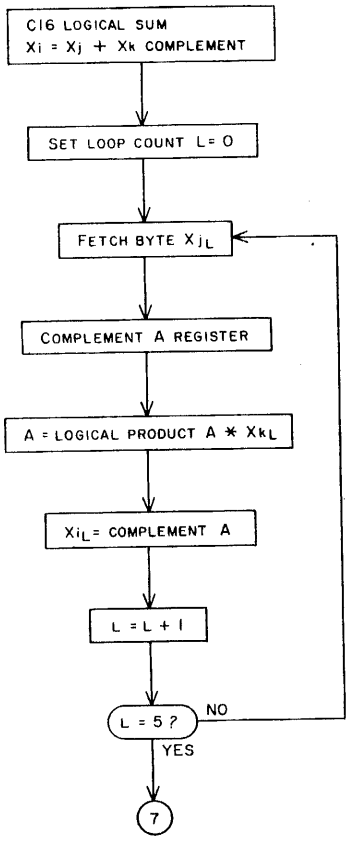

C11 LOGICAL PRODUCT
Xj · Xk → Xi
SET LOOP COUNT
L = 0
XjL (LP) XkL → XiL
L = L + 1
NO
L = 5 ?
YES
7

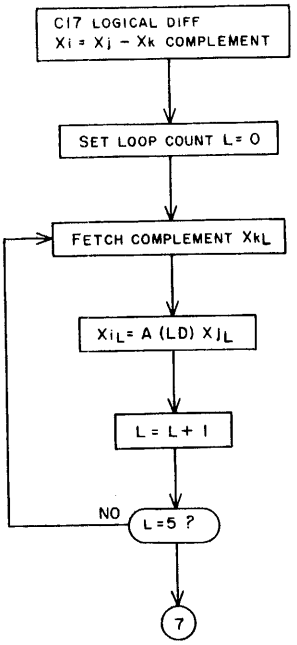

C12 LOGICAL SUM
Xj + Xk → Xi
SET LOOP COUNT
L = 0
SET A = COMPLEMENT
BYTE XjL
SET B = COMPLEMENT
BYTE XkL
SET C = A (LP) B
SET BYTE XiL =
COMPLEMENT C
L = L + 1
NO
L = 5 ?
YES
7

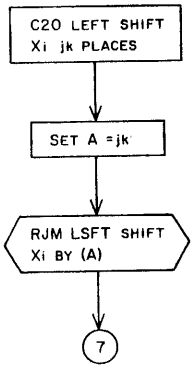

C13 LOGICAL DIFF.
Xj - Xk → Xi
SET LOOP COUNT
L = 0
BYTE
XiL = XjL (LD) XkL
L = L + 1
L = 5?
NO
7

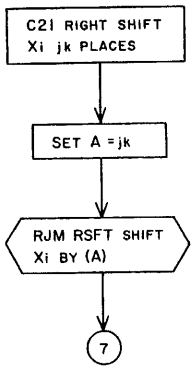

C14 TRANSMIT Xk
COMPLEMENT → Xi
SET LOOP COUNT
L = 0
FETCH BYTE XkL
TO A REGISTER
COMPLEMENT A
XiL = A
L = L + 1
L = 5?
NO
YES
7

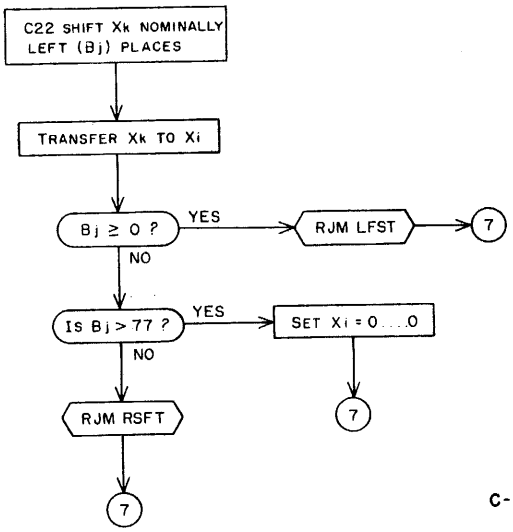

C15 LOGICAL PRODUCT
Xj · Xk COMPLEMENT → Xi
SET LOOP COUNT
L = 0
FETCH BYTE XkL
TO A REGISTER
COMPLEMENT A
XiL = XjL (LP) A
L = L + 1
NO
L = 5?
YES
7

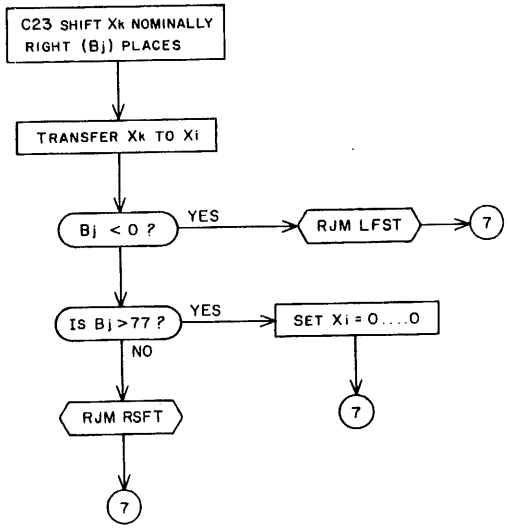

C16 LOGICAL SUM
Xi = Xj + Xk COMPLEMENT
SET LOOP COUNT L= 0
FETCH BYTE XjL
COMPLEMENT A REGISTER
A = LOGICAL PRODUCT A * XkL
XiL= COMPLEMENT A
L = L + 1
L = 5 ?
NO
YES
7

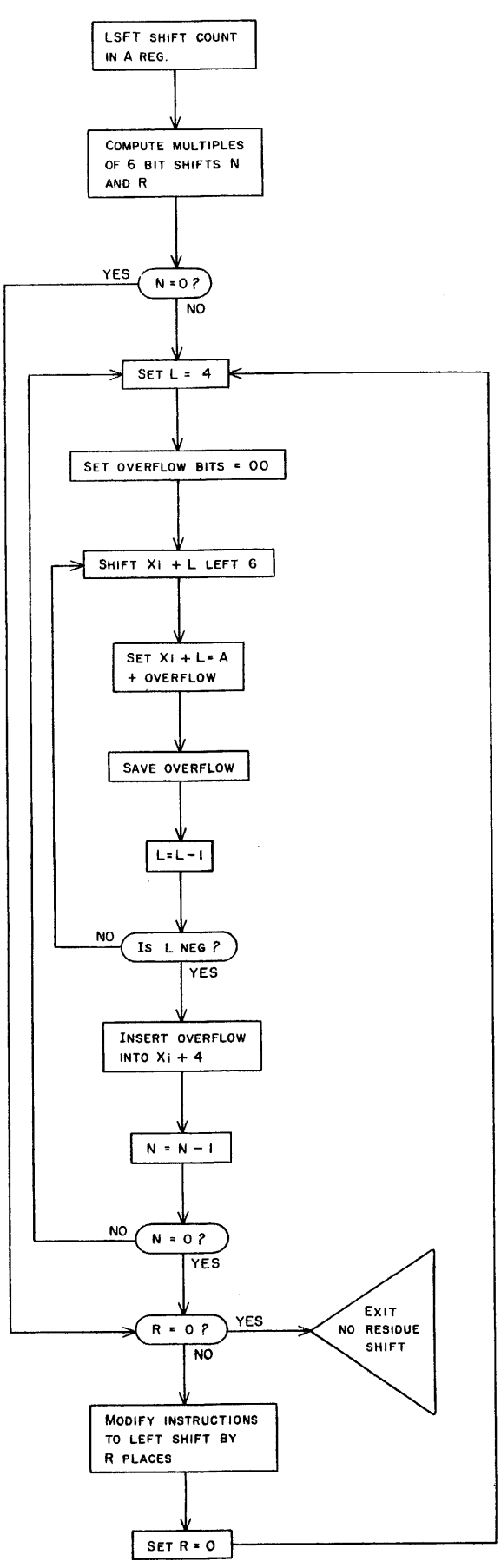

C17 LOGICAL DIFF
Xi = Xj - Xk COMPLEMENT
SET LOOP COUNT L= 0
FETCH COMPLEMENT XkL
XiL= A (LD) XjL
L = L + 1
NO
L=5 ?
7

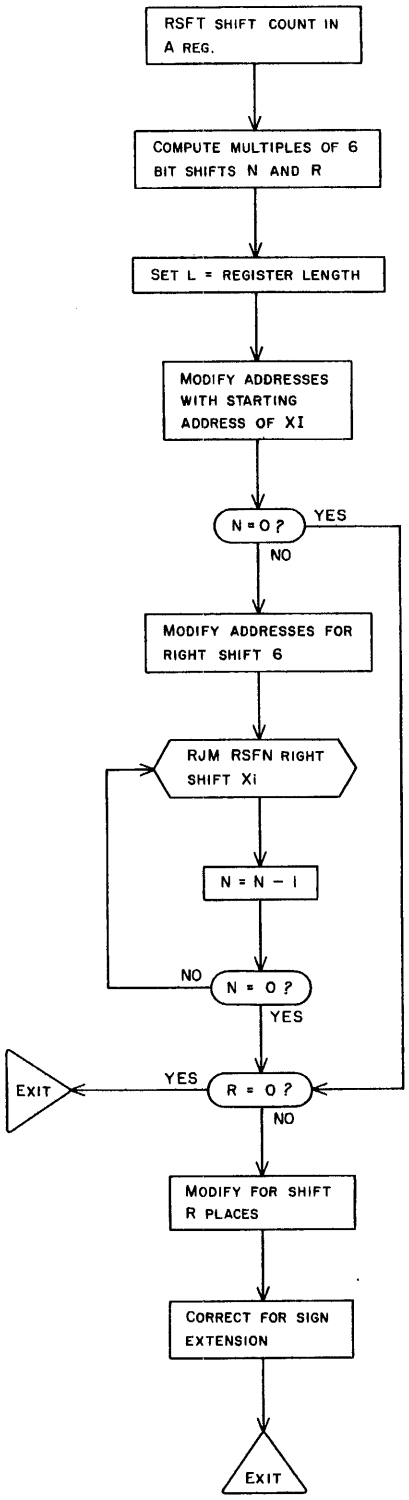

C20 LEFT SHIFT
Xi jk PLACES
SET A =jk
RJM LSFT SHIFT
Xi BY (A)
7

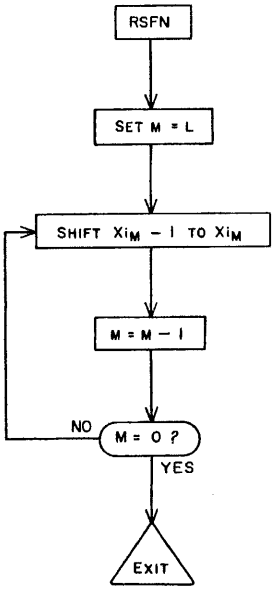

C21 RIGHT SHIFT
Xi jk PLACES
SET A =jk
RJM RSFT SHIFT
Xi BY (A)
7

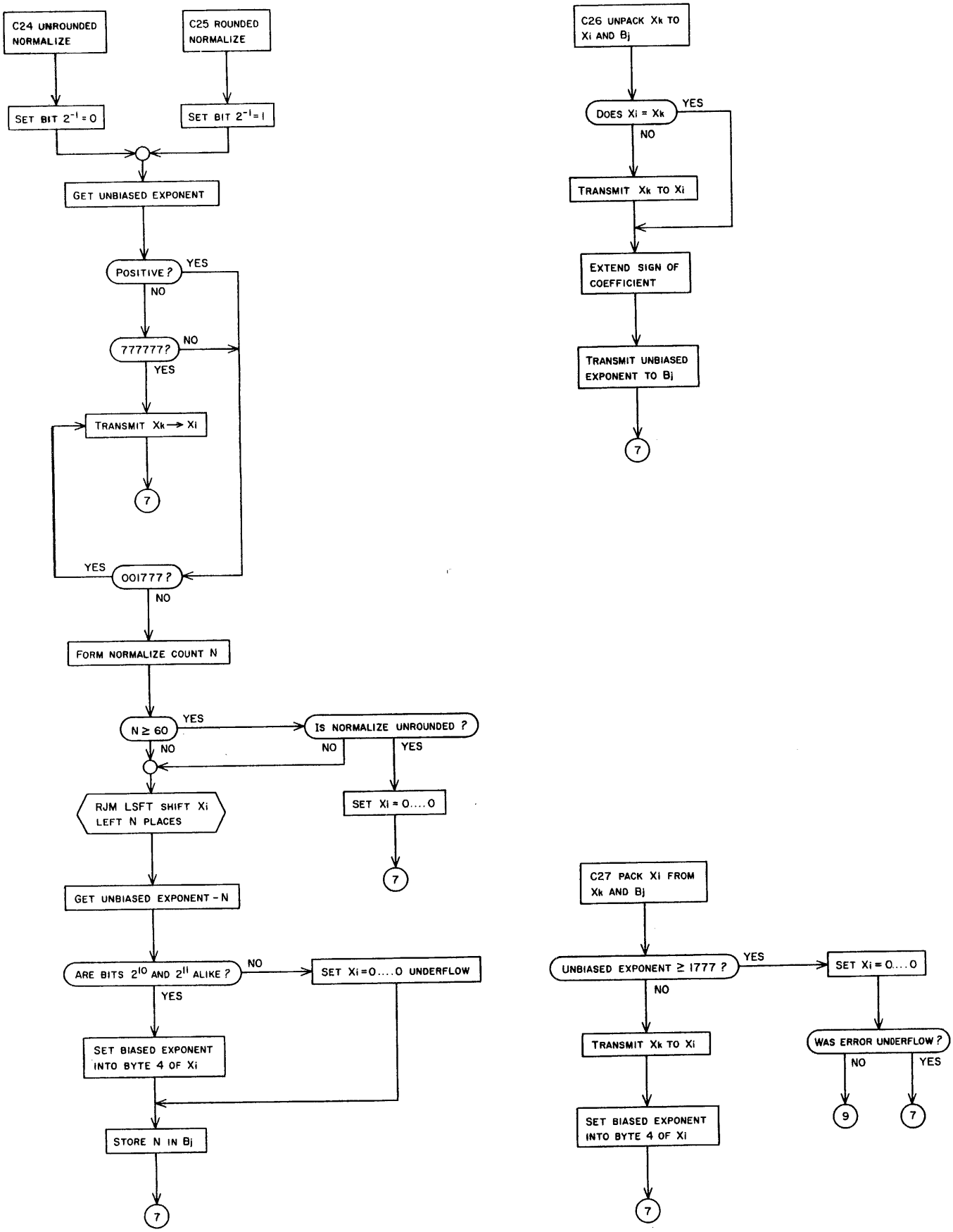

C22 SHIFT Xk NOMINALLY
LEFT (Bj) PLACES
TRANSFER Xk TO Xi
Bj ≥ 0 ?
YES
RJM LFST
7
NO
IS Bj > 77 ?
YES
SET Xi = 0....0
7
NO
RJM RSFT
7

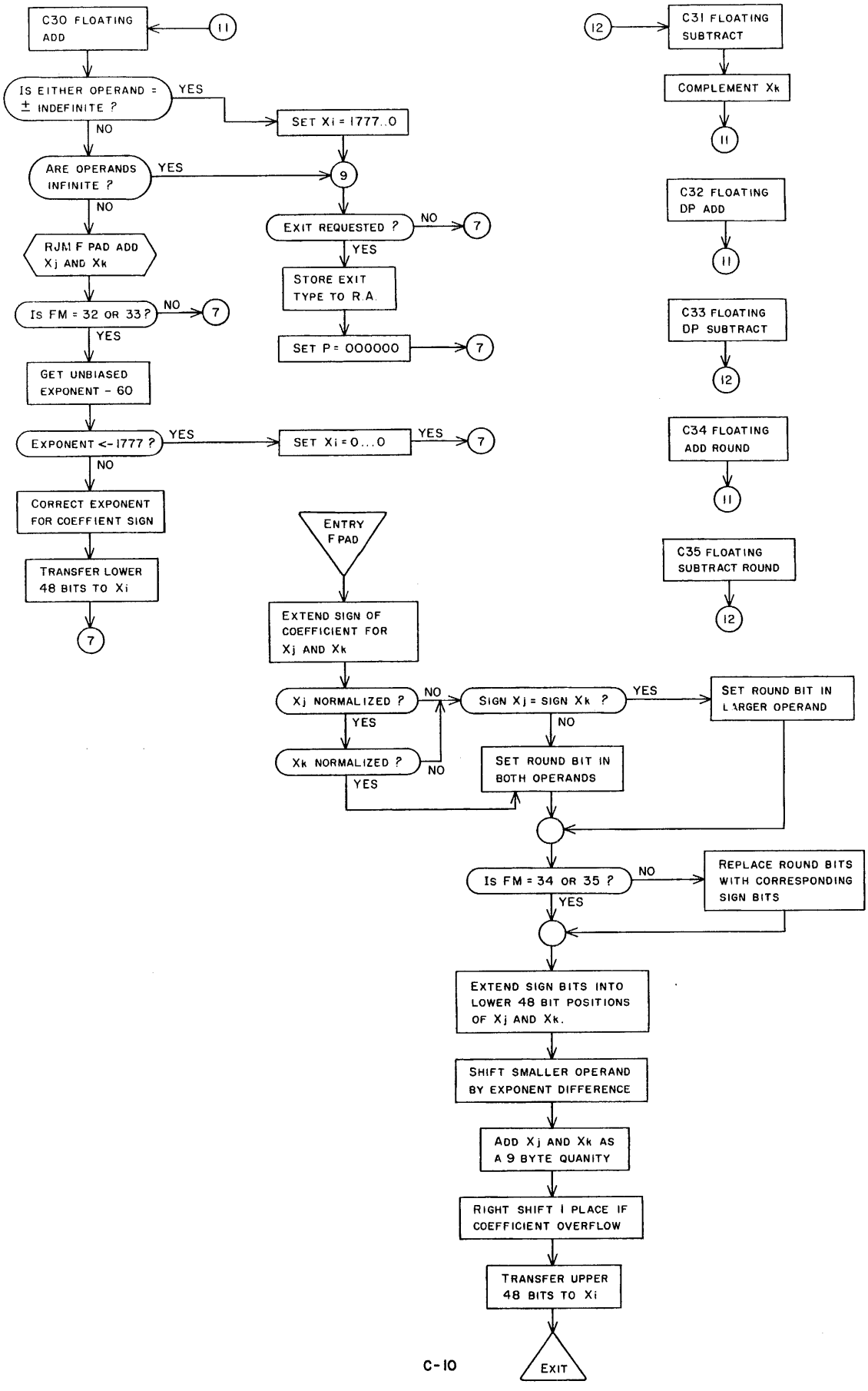

C23 SHIFT Xk NOMINALLY
RIGHT (Bj) PLACES
TRANSFER Xk TO Xi
Bj < 0 ?
YES
RJM LFST
7
IS Bj > 77 ?
YES
SET Xi = 0....0
7
NO
RJM RSFT
7

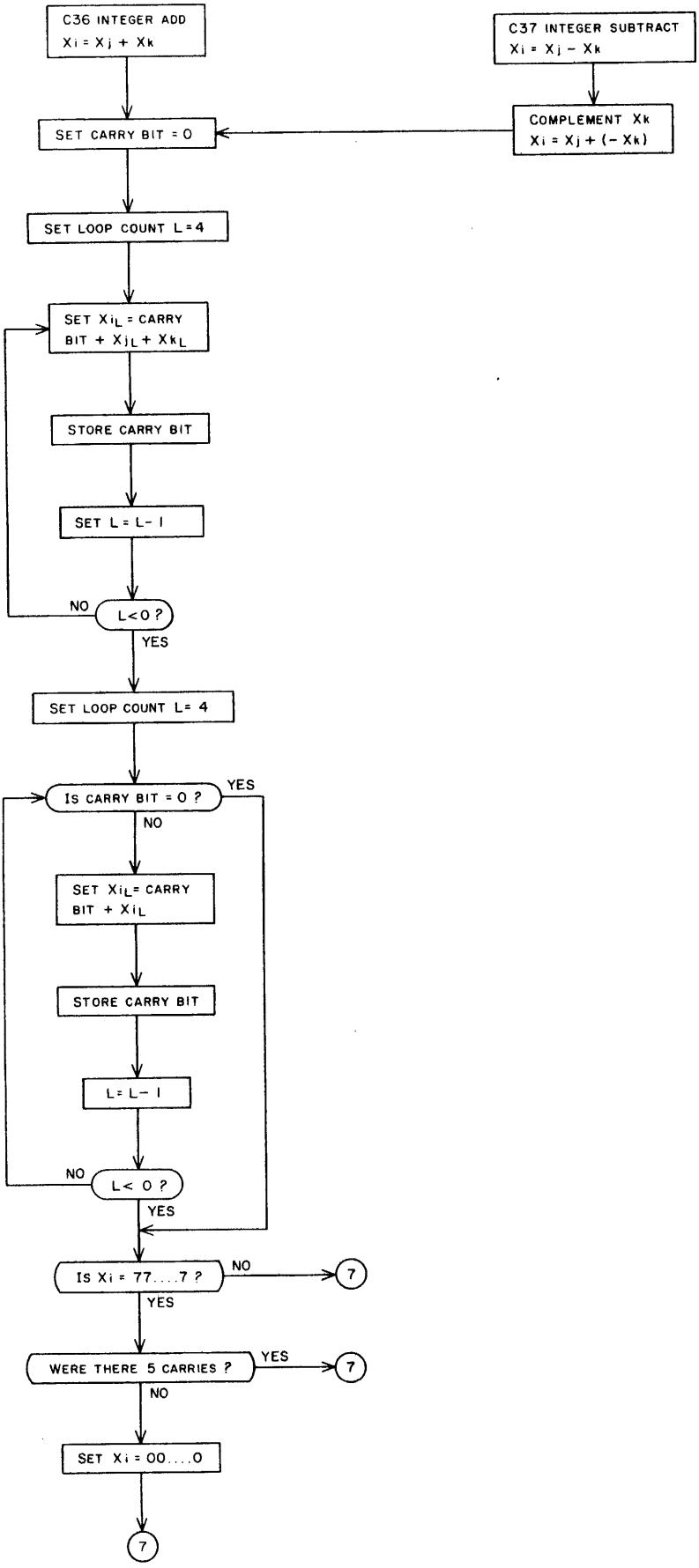

LSFT SHIFT COUNT IN A REG.
COMPUTE MULTIPLES OF 6 BIT SHIFTS N AND R
YES
N = 0 ?
NO
SET L = 4
SET OVERFLOW BITS = 00
SHIFT Xi + L LEFT 6
SET Xi + L = A + OVERFLOW
SAVE OVERFLOW
L=L-1
NO
IS L NEG ?
YES
INSERT OVERFLOW INTO Xi + 4
N = N - 1
NO
N = 0 ?
YES
R = 0 ?
YES
EXIT NO RESIDUE SHIFT
NO
MODIFY INSTRUCTIONS TO LEFT SHIFT BY R PLACES
SET R = 0

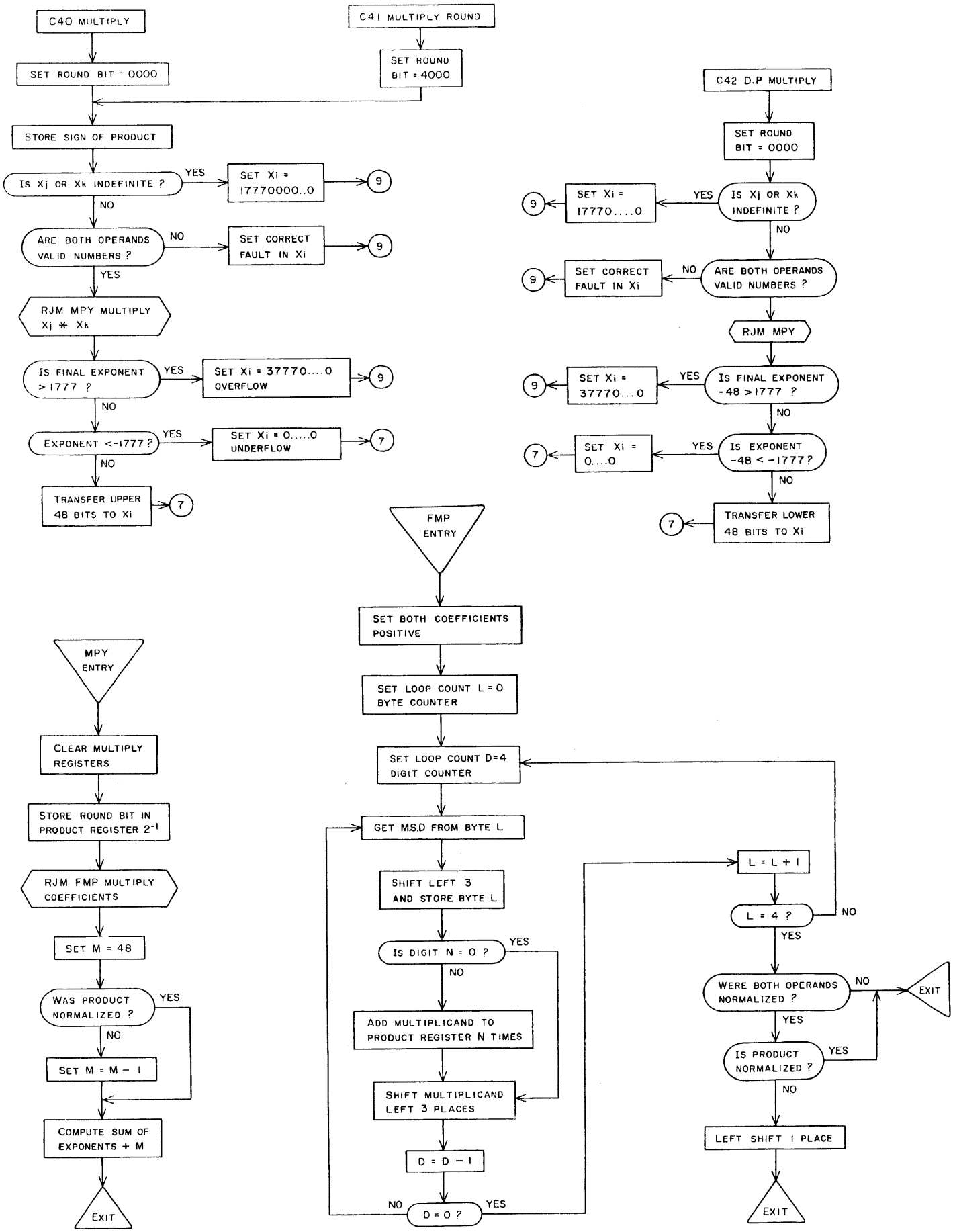

RSFT SHIFT COUNT IN A REG.
COMPUTE MULTIPLES OF 6 BIT SHIFTS N AND R
SET L = REGISTER LENGTH
MODIFY ADDRESSES WITH STARTING ADDRESS OF XI
N = 0 ?
YES
NO
MODIFY ADDRESSES FOR RIGHT SHIFT 6
RJM RSFN RIGHT SHIFT Xi
N = N - 1
NO
N = 0 ?
YES
EXIT
YES
R = 0 ?
NO
MODIFY FOR SHIFT R PLACES
CORRECT FOR SIGN EXTENSION
EXIT

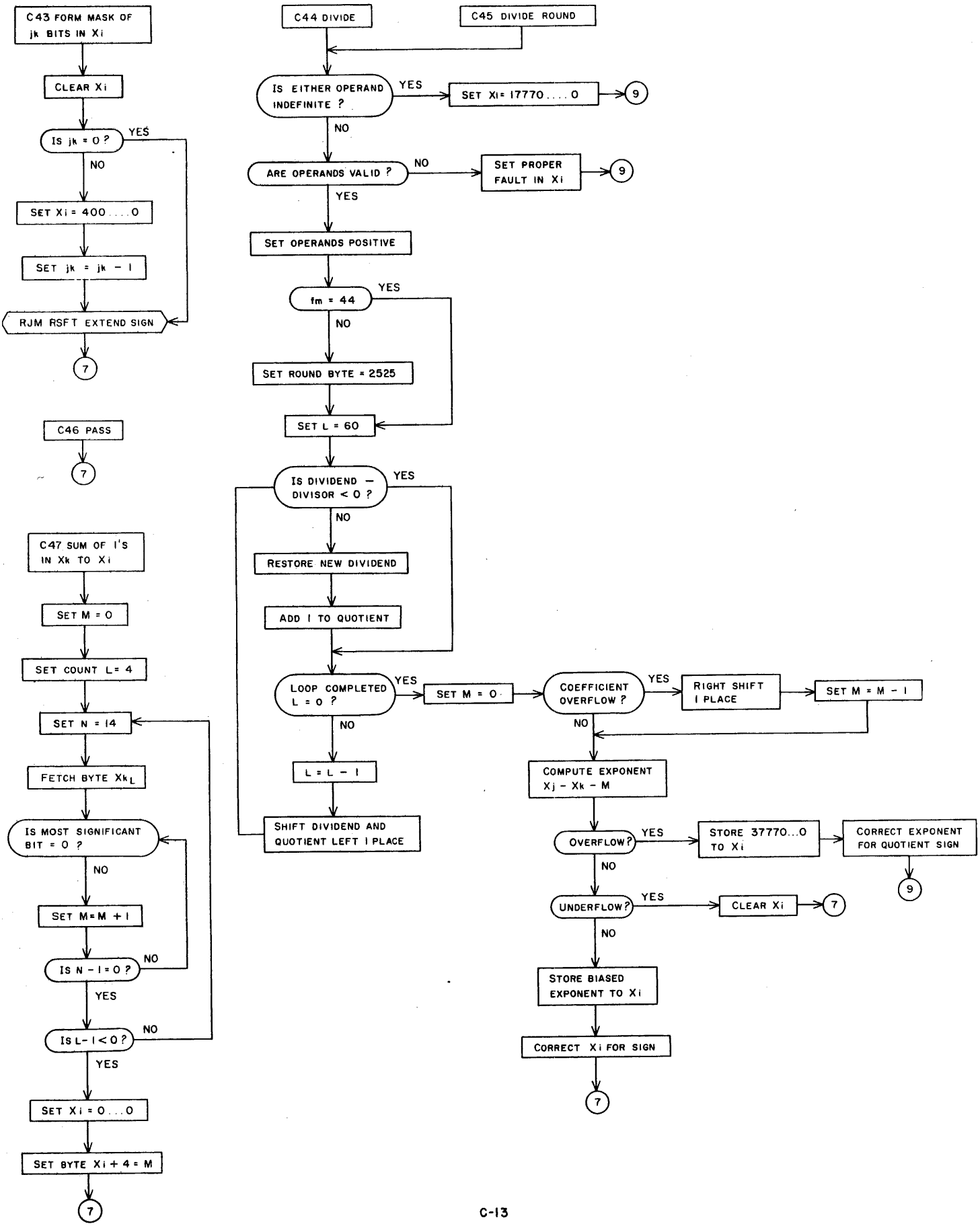

RSFN
SET M = L
SHIFT Xi_M - 1 TO Xi_M
M = M - 1
NO
M = 0 ?
YES
EXIT

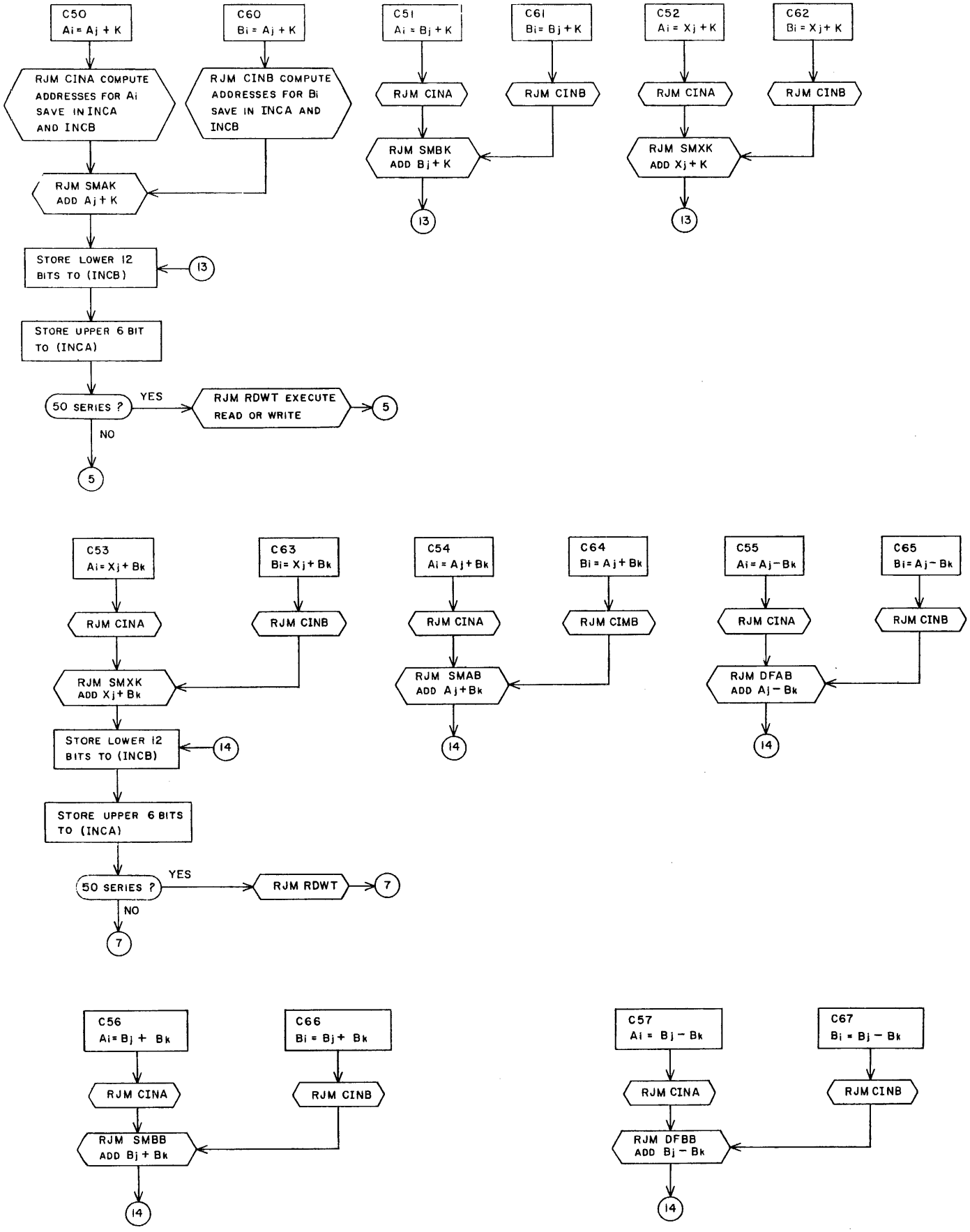

C24 UNROUNDED NORMALIZE
C25 ROUNDED NORMALIZE
SET BIT 2^-1 = 0
SET BIT 2^-1 = 1
GET UNBIASED EXPONENT
POSITIVE?
YES
NO
777777?
NO
YES
TRANSMIT Xk → Xi
7
YES
001777?
NO
FORM NORMALIZE COUNT N
N ≥ 60
YES
NO
IS NORMALIZE UNROUNDED ?
NO
YES
SET Xi = 0....0
7
RJM LSFT SHIFT Xi LEFT N PLACES
GET UNBIASED EXPONENT - N
ARE BITS 2^10 AND 2^11 ALIKE ?
NO
SET Xi=0....0 UNDERFLOW
YES
SET BIASED EXPONENT INTO BYTE 4 OF Xi
STORE N IN Bj
7
C26 UNPACK Xk TO Xi AND Bj
DOES Xi = Xk
YES
NO
TRANSMIT Xk TO Xi
EXTEND SIGN OF COEFFICIENT
TRANSMIT UNBIASED EXPONENT TO Bj
7
C27 PACK Xi FROM Xk AND Bj
UNBIASED EXPONENT ≥ 1777 ?
YES
NO
SET Xi = 0....0
TRANSMIT Xk TO Xi
WAS ERROR UNDERFLOW ?
NO
YES
SET BIASED EXPONENT INTO BYTE 4 OF Xi
9
7
7

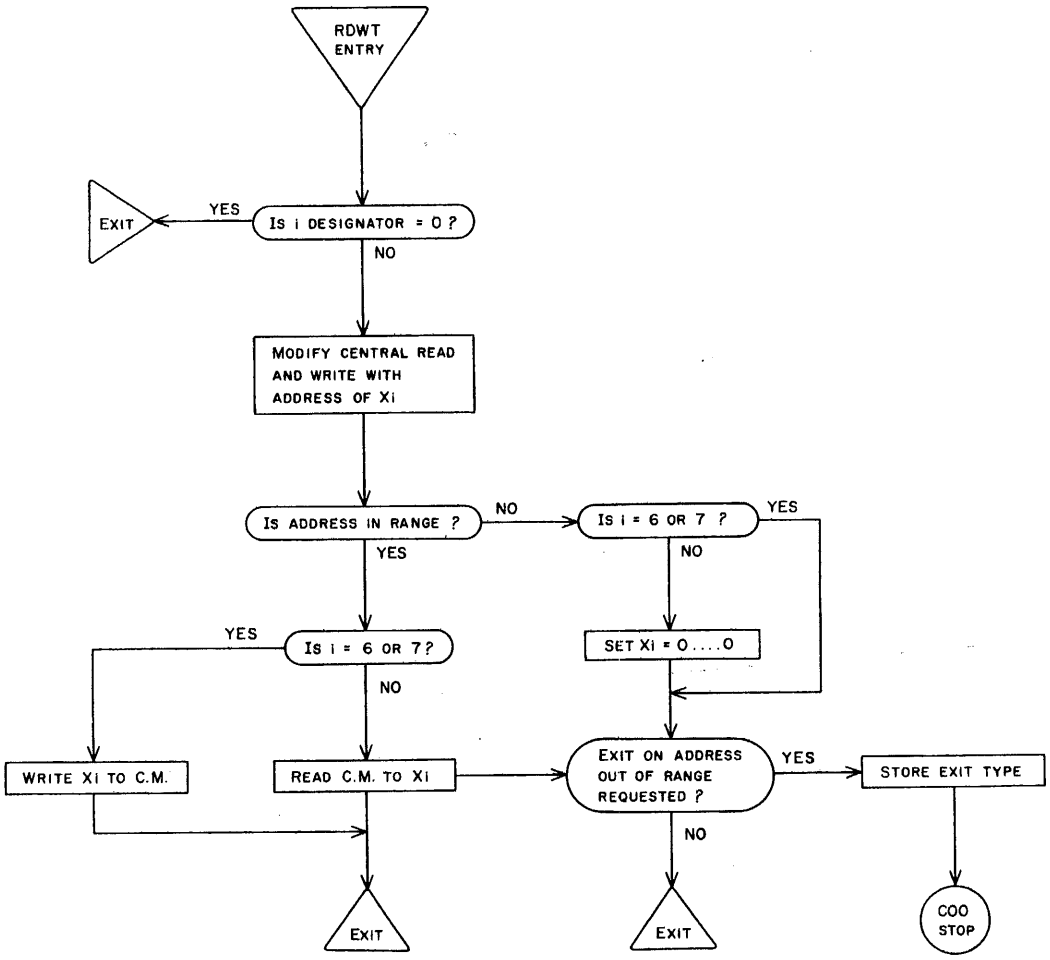

C30 FLOATING ADD
11
IS EITHER OPERAND = ± INDEFINITE ?
YES
NO
SET Xi = 1777..0
ARE OPERANDS INFINITE ?
YES
NO
9
EXIT REQUESTED ?
NO
7
YES
STORE EXIT TYPE TO R.A.
SET P = 000000
7
RJM F PAD ADD Xj AND Xk
IS FM = 32 OR 33 ?
NO
7
YES
GET UNBIASED EXPONENT - 60
EXPONENT <- 1777 ?
YES
SET Xi = 0...0
YES
7
NO
CORRECT EXPONENT FOR COEFFIENT SIGN
TRANSFER LOWER 48 BITS TO Xi
7
ENTRY F PAD
EXTEND SIGN OF COEFFICIENT FOR Xj AND Xk
Xj NORMALIZED ?
NO
YES
Xk NORMALIZED ?
NO
YES
SIGN Xj = SIGN Xk ?
YES
NO
SET ROUND BIT IN LARGER OPERAND
SET ROUND BIT IN BOTH OPERANDS
IS FM = 34 OR 35 ?
NO
YES
REPLACE ROUND BITS WITH CORRESPONDING SIGN BITS
EXTEND SIGN BITS INTO LOWER 48 BIT POSITIONS OF Xj AND Xk.
SHIFT SMALLER OPERAND BY EXPONENT DIFFERENCE
ADD Xj AND Xk AS A 9 BYTE QUANITY
RIGHT SHIFT 1 PLACE IF COEFFICIENT OVERFLOW
TRANSFER UPPER 48 BITS TO Xi
EXIT
12
C31 FLOATING SUBTRACT
COMPLEMENT Xk
11
C32 FLOATING DP ADD
11
C33 FLOATING DP SUBTRACT
12
C34 FLOATING ADD ROUND
11
C35 FLOATING SUBTRACT ROUND
12

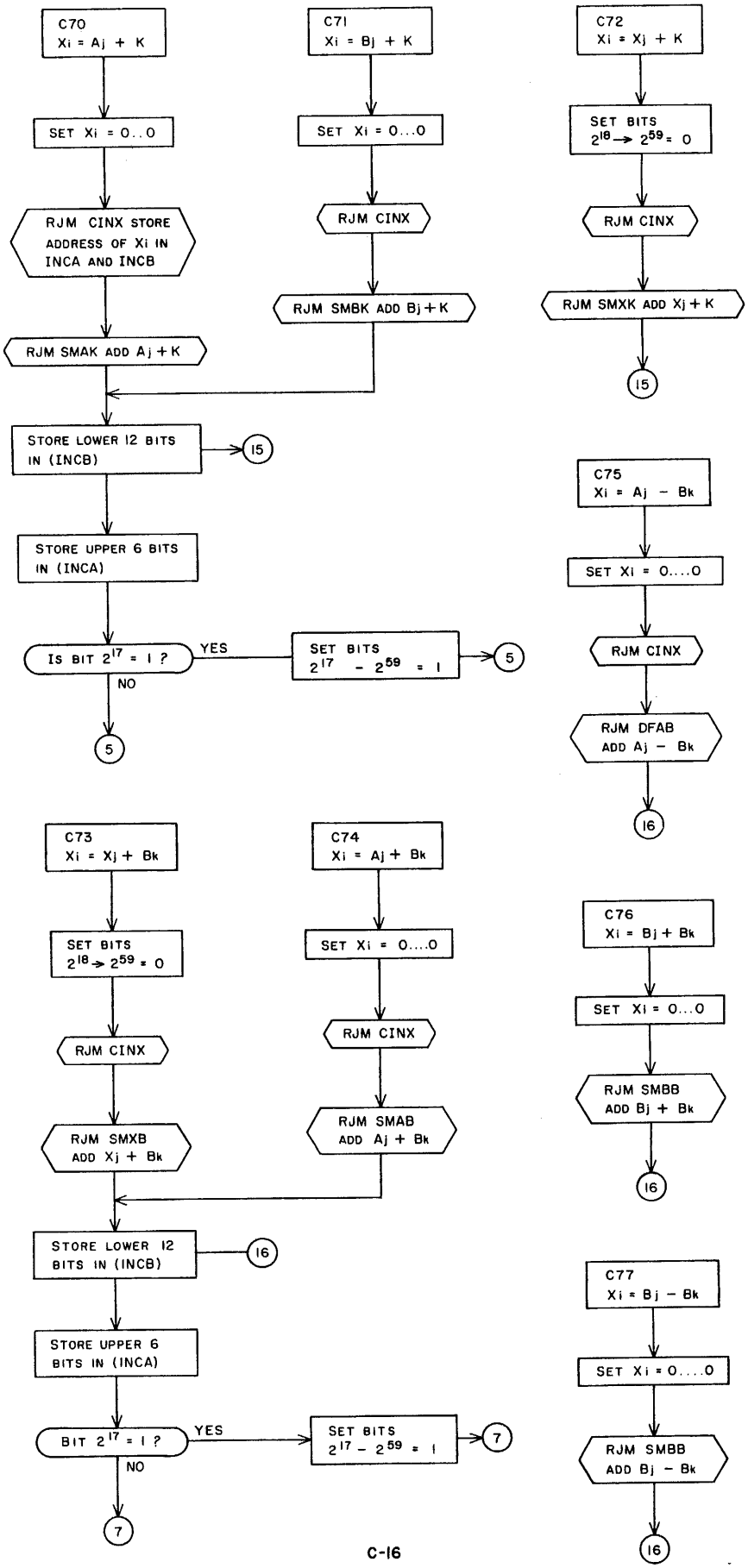

C36 INTEGER ADD
Xi = Xj + Xk
C37 INTEGER SUBTRACT
Xi = Xj - Xk
COMPLEMENT Xk
Xi = Xj + (- Xk)
SET CARRY BIT = 0
SET LOOP COUNT L=4
SET XiL = CARRY BIT + XjL + XkL
STORE CARRY BIT
SET L = L- 1
NO
L<0 ?
YES
SET LOOP COUNT L= 4
IS CARRY BIT = 0 ?
YES
NO
SET XiL= CARRY BIT + XiL
STORE CARRY BIT
L= L- 1
NO
L< 0 ?
YES
IS Xi = 77....7 ?
NO
7
YES
WERE THERE 5 CARRIES ?
YES
7
NO
SET Xi = 00....0
7


C40 MULTIPLY
SET ROUND BIT = 0000
C41 MULTIPLY ROUND
SET ROUND BIT = 4000
STORE SIGN OF PRODUCT
IS Xj OR Xk INDEFINITE ?
YES
SET Xi = 17770000..0
9
NO
ARE BOTH OPERANDS VALID NUMBERS ?
NO
SET CORRECT FAULT IN Xi
9
YES
RJM MPY MULTIPLY Xj * Xk
IS FINAL EXPONENT > 1777 ?
YES
SET Xi = 37770....0 OVERFLOW
9
NO
EXPONENT <-1777 ?
YES
SET Xi = 0.....0 UNDERFLOW
7
NO
TRANSFER UPPER 48 BITS TO Xi
7
C42 D.P MULTIPLY
SET ROUND BIT = 0000
IS Xj OR Xk INDEFINITE ?
YES
SET Xi = 17770....0
9
NO
ARE BOTH OPERANDS VALID NUMBERS ?
NO
SET CORRECT FAULT IN Xi
9
RJM MPY
IS FINAL EXPONENT -48 > 1777 ?
YES
SET Xi = 37770...0
9
NO
IS EXPONENT -48 < -1777 ?
YES
SET Xi = 0....0
7
NO
TRANSFER LOWER 48 BITS TO Xi
7
FMP ENTRY
SET BOTH COEFFICIENTS POSITIVE
SET LOOP COUNT L = 0 BYTE COUNTER
SET LOOP COUNT D=4 DIGIT COUNTER
GET M.S.D FROM BYTE L
SHIFT LEFT 3 AND STORE BYTE L
IS DIGIT N = 0 ?
YES
NO
ADD MULTIPLICAND TO PRODUCT REGISTER N TIMES
SHIFT MULTIPLICAND LEFT 3 PLACES
D = D - 1
D = 0 ?
NO
YES
L = L + 1
L = 4 ?
NO
YES
WERE BOTH OPERANDS NORMALIZED ?
NO
EXIT
YES
IS PRODUCT NORMALIZED ?
YES
NO
LEFT SHIFT 1 PLACE
EXIT
MPY ENTRY
CLEAR MULTIPLY REGISTERS
STORE ROUND BIT IN PRODUCT REGISTER 2-1
RJM FMP MULTIPLY COEFFICIENTS
SET M = 48
WAS PRODUCT NORMALIZED ?
YES
NO
SET M = M - 1
COMPUTE SUM OF EXPONENTS + M
EXIT


C43 FORM MASK OF jk BITS IN Xi
CLEAR Xi
IS jk = 0 ?
YES
NO
SET Xi = 400....0
SET jk = jk - 1
RJM RSFT EXTEND SIGN
7
C46 PASS
7
C47 SUM OF 1'S IN Xk TO Xi
SET M = 0
SET COUNT L= 4
SET N = 14
FETCH BYTE Xk_L
IS MOST SIGNIFICANT BIT = 0 ?
NO
SET M=M + 1
IS N - 1 = 0 ?
NO
YES
IS L-1<0 ?
NO
YES
SET Xi = 0...0
SET BYTE Xi + 4 = M
7
C44 DIVIDE
C45 DIVIDE ROUND
IS EITHER OPERAND INDEFINITE ?
YES
SET Xi= 17770....0
9
NO
ARE OPERANDS VALID ?
NO
SET PROPER FAULT IN Xi
9
YES
SET OPERANDS POSITIVE
fm = 44
YES
NO
SET ROUND BYTE = 2525
SET L = 60
IS DIVIDEND - DIVISOR < 0 ?
YES
NO
RESTORE NEW DIVIDEND
ADD 1 TO QUOTIENT
LOOP COMPLETED L = 0 ?
YES
SET M = 0
NO
L = L - 1
SHIFT DIVIDEND AND QUOTIENT LEFT 1 PLACE
COEFFICIENT OVERFLOW ?
YES
RIGHT SHIFT 1 PLACE
SET M = M - 1
NO
COMPUTE EXPONENT Xj - Xk - M
OVERFLOW?
YES
STORE 37770...0 TO Xi
CORRECT EXPONENT FOR QUOTIENT SIGN
9
NO
UNDERFLOW?
YES
CLEAR Xi
7
NO
STORE BIASED EXPONENT TO Xi
CORRECT Xi FOR SIGN
7


C50
Ai = Aj + K
C60
Bi = Aj + K
RJM CINA COMPUTE ADDRESSES FOR Ai SAVE IN INCA AND INCB
RJM CINB COMPUTE ADDRESSES FOR Bi SAVE IN INCA AND INCB
RJM SMAK
ADD Aj + K
STORE LOWER 12 BITS TO (INCB)
13
STORE UPPER 6 BIT TO (INCA)
50 SERIES ?
YES
RJM RDWT EXECUTE READ OR WRITE
5
NO
5
C51
Ai = Bj + K
C61
Bi = Bj + K
RJM CINA
RJM CINB
RJM SMBK
ADD Bj + K
13
C52
Ai = Xj + K
C62
Bi = Xj + K
RJM CINA
RJM CINB
RJM SMXK
ADD Xj + K
13
C53
Ai = Xj + Bk
C63
Bi = Xj + Bk
RJM CINA
RJM CINB
RJM SMXK
ADD Xj + Bk
STORE LOWER 12 BITS TO (INCB)
14
STORE UPPER 6 BITS TO (INCA)
50 SERIES ?
YES
RJM RDWT
7
NO
7
C54
Ai = Aj + Bk
C64
Bi = Aj + Bk
RJM CINA
RJM CIMB
RJM SMAB
ADD Aj + Bk
14
C55
Ai = Aj - Bk
C65
Bi = Aj - Bk
RJM CINA
RJM CINB
RJM DFAB
ADD Aj - Bk
14
C56
Ai = Bj + Bk
C66
Bi = Bj + Bk
RJM CINA
RJM CINB
RJM SMBB
ADD Bj + Bk
14
C57
Ai = Bj - Bk
C67
Bi = Bj - Bk
RJM CINA
RJM CINB
RJM DFBB
ADD Bj - Bk
14


RDWT
ENTRY
EXIT
YES
Is i DESIGNATOR = 0 ?
NO
MODIFY CENTRAL READ
AND WRITE WITH
ADDRESS OF Xi
Is ADDRESS IN RANGE ?
NO
Is i = 6 OR 7 ?
YES
YES
NO
YES
Is i = 6 OR 7?
SET Xi = 0....0
NO
WRITE Xi TO C.M.
READ C.M. TO Xi
EXIT ON ADDRESS
OUT OF RANGE
REQUESTED ?
YES
STORE EXIT TYPE
NO
EXIT
EXIT
COO
STOP


C70
Xi = Aj + K
SET Xi = 0...0
RJM CINX STORE ADDRESS OF Xi IN INCA AND INCB
RJM SMAK ADD Aj + K
C71
Xi = Bj + K
SET Xi = 0...0
RJM CINX
RJM SMBK ADD Bj + K
C72
Xi = Xj + K
SET BITS 2^18 → 2^59 = 0
RJM CINX
RJM SMXK ADD Xj + K
15
STORE LOWER 12 BITS IN (INCB)
15
STORE UPPER 6 BITS IN (INCA)
IS BIT 2^17 = 1 ?
YES
SET BITS 2^17 - 2^59 = 1
5
NO
5
C75
Xi = Aj - Bk
SET Xi = 0....0
RJM CINX
RJM DFAB ADD Aj - Bk
16
C73
Xi = Xj + Bk
SET BITS 2^18 → 2^59 = 0
RJM CINX
RJM SMXB ADD Xj + Bk
C74
Xi = Aj + Bk
SET Xi = 0....0
RJM CINX
RJM SMAB ADD Aj + Bk
C76
Xi = Bj + Bk
SET Xi = 0...0
RJM SMBB ADD Bj + Bk
16
STORE LOWER 12 BITS IN (INCB)
16
STORE UPPER 6 BITS IN (INCA)
BIT 2^17 = 1 ?
YES
SET BITS 2^17 - 2^59 = 1
7
NO
7
C77
Xi = Bj - Bk
SET Xi = 0....0
RJM SMBB ADD Bj - Bk
16

CUT ALONG LINE

COMMENT SHEET

6600 CHIPPEWA OPERATING SYSTEM

Pub. No. 60124500

FROM NAME : _____

BUSINESS ADDRESS : _____

COMMENTS: (DESCRIBE ERRORS, SUGGESTED ADDITION OR DELETION AND INCLUDE PAGE NUMBER, ETC.)

NO POSTAGE STAMP NECESSARY IF MAILED IN U. S. A.

FOLD ON DOTTED LINES AND STAPLE

STAPLE

STAPLE

FOLD

FOLD

FIRST CLASS
PERMIT NO. 8241

MINNEAPOLIS, MINN.

BUSINESS REPLY MAIL

NO POSTAGE STAMP NECESSARY IF MAILED IN U.S.A.

POSTAGE WILL BE PAID BY

CONTROL DATA CORPORATION
8100 34TH AVENUE SOUTH
MINNEAPOLIS 20, MINNESOTA

ATTN: TECHNICAL PUBLICATIONS DEPT.
COMPUTER DIVISION
PLANT TWO

CUT ALONG LINE

FOLD

FOLD

STAPLE

STAPLE