CONTROL DATA CORPORATION

Development Division - Applications

ALPHABETIC PERIPHERAL PACKAGES

Chippewa Operating System

10/20/65

Table of Contents

ALPHABETIC PERIPHERAL PACKAGES

INTRODUCTION

The packages described on the following pages may be called by a central program. They are loaded into a peripheral processor from either RPL (resident peripheral library) or PLD (peripheral library directory). A central program, by setting the package name in left-justified display code in RA+1, requests MTR to assign the package to a free PP. Each package begins execution at location 1000 in the PP and arguments are passed to it from the central program through the lower portion of RA+1. If the execution of the package is terminated normally or abnormally the PP is released and must be reassigned when it is needed again.

The last section of this narrative gives a few practical examples about the use of some of the routines.

ROUTINE: DMP -- Storage Dump

PURPOSE: To enter an octal dump of a requested area of central memory into the OUTPUT file.

GENERAL: This package may be called by a control card or DIS console. Three calls may be made:

a) no parameters - dump only exchange package

b) one argument - dump from RA to the specified address

c) two arguments - dump area between the two addresses

METHOD:

1. Two checks made on the arguments passed through the input register may cause a diagnostic:

   a) terminal address < initial address

   b) terminal address > field length

   Either condition will cause a "DMP ARG ERROR" dayfile message and the control point aborted.

2. In the case that both parameters are equal, i.e., usually zero, the exchange jump area (first 16 words of control point area) is set up as the dump address. The title of the dump is changed to "DMPX."

3. The FNT is searched for a file of local or common and assigned to this control point. The name must be OUTPUT and the file on disk 0 with buffer status indicating not busy (odd value). If no such file is found, an entry of this type is made into FNT so that the dump can be printed. The file status in either case is set to 14_8 (request coded write).

4. If no OUTPUT file was found while searching FNT, then the new file just added must have a track assignment. A track is requested of MTR and when it is assigned the number is inserted in the FST entry of the new file.

5. If the last reference to the file was a read operation, then no dumping will be done. This prevents writing over output data that may have been repositioned by the read.

6. The dump has a header of either DMPX, for an exchange area dump, or DMP. for any other dump. Each central memory word has an address relative to RA and 4 five digit groups of data with two spaces between the address and data and a space separating each byte. The peripheral buffer spans from 2000-7000 and is filled before it is passed to the output file.

7. The dump address is incremented by one until the terminal address specified in the input register is reached. A return jump is issued to dump the PP buffer into the OUTPUT file when it is full or the terminal address is encountered.

8. The PP buffer is written on disk 0 a full sector (100_8 words) at a time until short sector is found. It is written on the disk followed by a file mark and then channel 0 is released. The buffer input address is reset so that more data may be inserted if the terminal address has not been reached. Every write to the disk is terminated by a file mark but the sector number is not incremented. This will prevent a file from ever running away but still allow more information to erase the file mark and reside within one file.

9. After the formatted octal dump has been successfully passed to the OUTPUT file, the buffer status byte in FST is changed to 15_8 (completed coded write). Then the PP is released.

NOTES:

1. Successive identical lines are not suppressed.

2. One print line contains only an address and a central memory word.

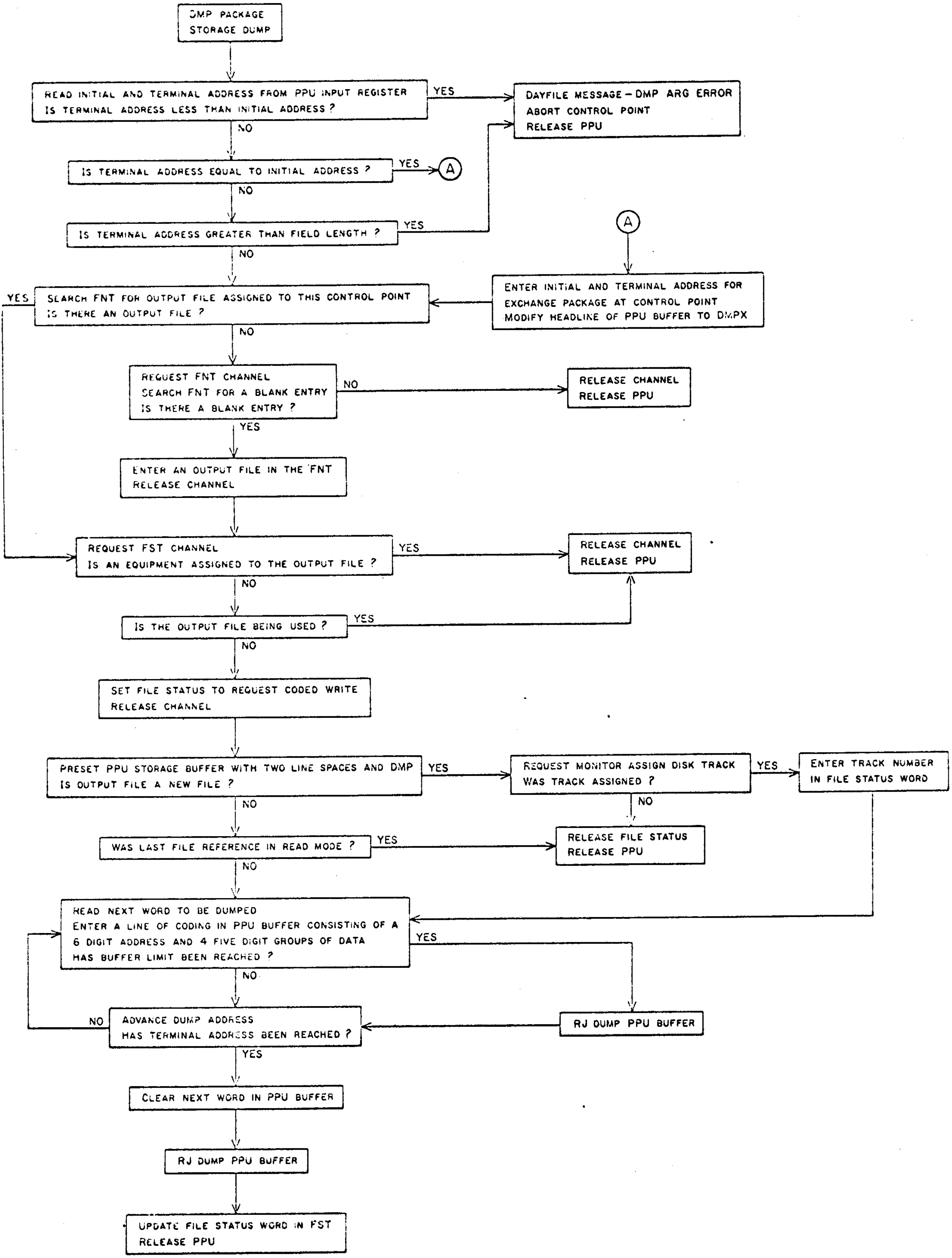
DMP Routines

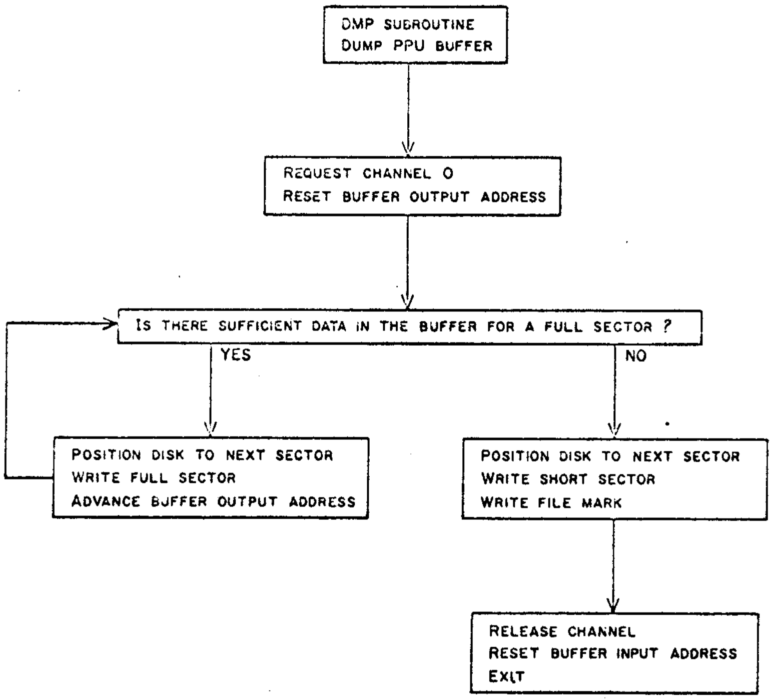
1000	Main Program	1560, 1100, 1200, 12-760, 100, 530, 13-760
1100	Search for Output File	740, 750, 12-760, 100
1200	Enter Output File	6-760, 1300, 1600
1300	Enter Line in Buffer	1600
1560	Process Exchange Area	
1600	Dump Buffer	740, 1700, 700, 1740, 750
1700	Enter Control Byte	6-760
1740	Write Sector	
2000	Disk Buffer	
2030	Begin Output File	15-740, 750, 12-760, 100

Direct Core Cells

1000	P10/14	CP Status
	P50/54	Input register contents
	P55	RA
	P60/61	First argument
	P62/63	Second argument
	P74	CP address
	P75	Input register address
1100	P01	File type (local or common)
	P10/14	FNT entry
	P20/24	FNT status later FST entry
	P45	Last buffer status from FST
	P50/54	Input register contents
	P57	FST address
1200	P01	Central memory word count
	P10/14	PP message buffer contents
	P20/24	FST entry
	P45	Last buffer status from FST
	P57	FST address
	P60/61	First argument

	P62/63	Second argument
	P64	IN address for PP buffer
1300	P10/14	Central memory word to be dumped
	P60/61	First argument
	P64	IN address for PP buffer
1560	P60/61	First argument
	P62/63	Second argument
	P55	RA
	P74	CP address
1600	P01	Central memory word count
	P02	Sector length
	P20/24	FST entry
	P64	IN address for PP buffer
	P65	OUT address for PP buffer
1700	P02	Disk status byte
	P10/14	Message buffer
	P20/24	FST entry
1740	P01	Disk status byte from FST
	P20/24	FST entry
2030	P01	FNT index
	P10/14	FNT entry
	P20/24	FNT status


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
A
NO
ENTER INITIAL AND TERMINAL ADDRESS FOR
EXCHANGE PACKAGE AT CONTROL POINT
MODIFY HEADLINE OF PPU BUFFER TO DMPX
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
RELEASE FILE STATUS
RELEASE PPU
NO
WAS LAST FILE REFERENCE IN READ MODE ?
YES
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
RJ DUMP PPU BUFFER
NO
ADVANCE DUMP ADDRESS
HAS TERMINAL ADDRESS BEEN REACHED ?
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

ROUTINE EXU - Execute Compiled Program

PURPOSE To locate and read a specified file from the disk into central memory. The appropriate exchange jump package parameters are set up and then the central processor is told that the file is ready for execution.

GENERAL After a file has been compiled and stored on the disk, EXU is used to load a file into central memory beginning at the calling program's reference address. The location of the name of the file (left-justified display code) to be called and executed is set in the lower 18 bits of the input register.

METHOD

1. The error flag at the control point is checked. If it is set, the package is released so that error processing may proceed.

2. The file name is read in by adding RA and the lower 18 bits of the input register. FNT is searched for the file name and if it is located a check is made on its control point assignment.

3. When the file is located, its type from the FNT is checked for input and output. Only common or local files may be executed.

4. The FST entry must reflect that the file is on disk and has been used.

5. A dayfile message of "PROGRAM NOT ON DISK" is sent if:
   a) The file name was not located in the FNT.
   b) The file was not assigned to the calling control point.
   c) The file has either an input or output status.
   d) The file has an equipment other than disk assigned, i.e. it is a card file or tape file.
   e) The file has not been used, i.e. no track has been assigned. This status is reflected by checking the beginning track byte in the FST for non-zero.

6. A request for channel 0 is made and the disk is positioned to the beginning track and sector for the file.

7. The file is read and stored one sector at a time into central memory beginning at the control points' reference address. Encountering a short sector or reaching the field limit causes the reading of the file to be terminated.

8. If the field limit was reached before the end of the file, a dayfile message of "PROGRAM TOO LONG" appears and the control point aborted.

9. The exchange jump package in the control point area is updated to permit execution of the newly loaded program.

   a) First the sense lights and switches from word 26 of the control point are sored in RA.

   b) RA + 1 is read and then cleared.

   c) P in the exchange jump area is set to the number of parameters from RA+1 plus 3. The field length (in hundreds) from word 20 is stored in A_0.

10. The central processor is then requested by a MTR code 15_8. When this request has been processed, the PP is released.

NOTES

1. The file read in off the disk is loaded beginning at RA so that a portion or all of the calling program may be overlayed.

2. Sense lights and switches are passed from the calling program to the new program through RA.

3. The field length specified in RA of the called program is ignored. The field length assigned to the control point is checked.

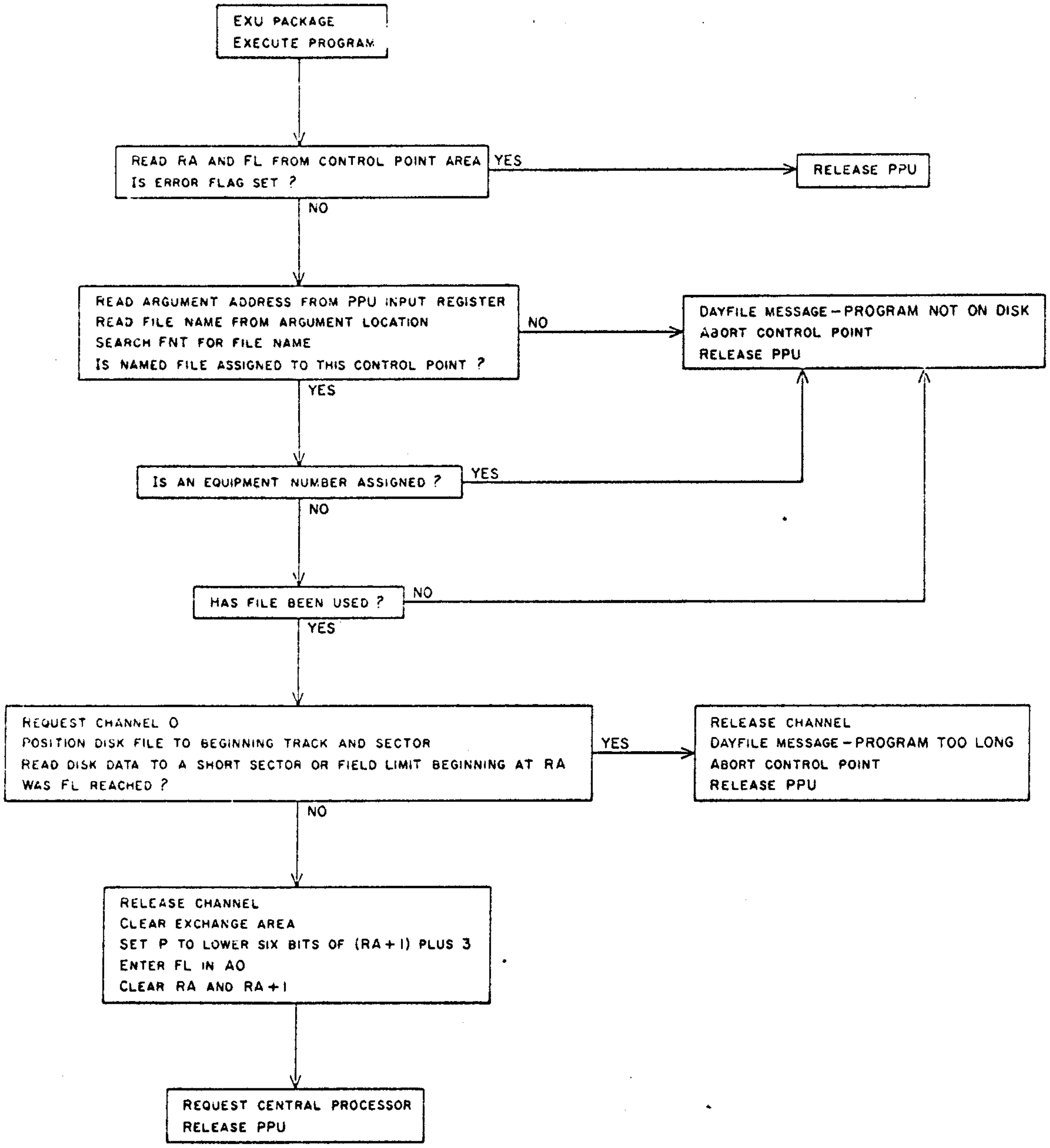
EXU Routines

1000	Main Program	12-760, 100, 1100, 1200, 1300, 15-760, 531, 13-760
1100	Search for file	1160, 1064
1200	Read program from disk	740, 700, 400, 750
1300	Clear exchange area	

Direct Core Cells

1000	P06	beginning track number of file
	P07	sector number
	P10/14	CP(20) - status word
	P20/24	contents of FST entry
	P50/54	contents of input register
	P55	RA
	P56	FL
	P57	FST status
	P74	control point address
	P75	address of input register
	P7200/7702	disk buffer
1100	P01	control point assignment
	P10/14	FNT entry, later FST entry
	P20/24	File name in left-justified display code
	P30/34	FNT status

1200	P01	control point assignment
	P04	RA (in hundreds)
	P05	FL (in hundreds)
	P06	track number
	P07	sector number
1300	P01	control point assignment
	P10/14	zeroed, later each word of exchange area
	P20/24	CP (26), later RA+1
	P30/34	CP (20)


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

ROUTINE: CLL -- Central Library Loader

PURPOSE: To load one or more overlays into an area specified by a central memory calling program.

GENERAL: The location (BA) of the overlay parameters is set into the lower 18 bits of the input register. The location (BA)

BA		FWA
BA+1		LIMIT
BA+2	NAME	ADDR
	⋮	
BA+n+2	0--------------------0	

FWA - beginning address for first overlay

LIMIT - last address for group of overlays

NAME - name of overlay (left-justified display code)

The address of where the overlay begins will be returned in the lower 18 bits of its location in the BA area (ADDR). If it cannot be loaded because the length exceeds LIMIT, an address of 777777_8 will be inserted. The address will remain cleared if the overlay cannot be located. A zero word must terminate the parameters.

METHOD:

1. The RA and FL are read from CP(20) and stored in hundreds.

2. From the input register the location of BA is read and incremented by RA so the first parameter is read.

3. When the LIMIT is read, a check is made to insure that it is within the field length. If LIMIT exceeds FL, the PP is released and no diagnostic results.

4. Each argument is read and checked for zero. If it is zero, then the list is assumed to be exhausted.

5. The resident subroutine library (RSL) is first searched. The first entry in RSL is checked against the name of the overlay. If a match is not found, then the field length of the RSL entry is added to the beginning address of RSL, in order to find the next subroutine in the table.

6. If the overlay is found in RSL, the length is added to FWA and that total may not exceed LIMIT. If it does, then 77777_8 is entered into the beginning address area of that overlay (ADDR). The next argument will be read and processed.

7. FWA will reflect the next available location for loading so it is stored as ADDR for that argument. The program is transferred 100_8 words at a time and FWA is increased by the number of words stored until a short record is encountered. A zero length record is not transmitted.

8. FWA is increased to the next available central memory address and then the next argument is processed.

9. If the name is not found in RSL then the central library directory (CLD) is searched, The format of this table is:

NAME (DISPLAY CODE)	SEC	TRACK
42	6	12

It is terminated by a zero word or table limit.

10. When the overlay is found to be in CLD, then FWA is stored as the overlay's beginning address. Channel 0 for the disk is requested and it is positioned to the proper track.

11. One sector at a time is read and its length recorded. If a zero length is found, it is not transmitted. If the sector length exceeds the number of words to LIMIT then they are not stored and 77777_8 is set as the beginning address. Track repositioning is checked after every read. If the short sector is encountered before the LIMIT exceeded, it is stored and FWA is updated to be the next available program address. Another argument is then processed.

12. If the name was not found in the RSL or CLD, then the job file is read. If the package is found in the FNT, then it must be assigned to the calling program's control point and be on disk 0. If it is not, then the next argument is processed.

13. When a file is found in the FNT, the same disk operations apply as those with CLD.

14. When an argument is found to be zero then the next available program address (FWA) is zeroed. BA is also cleared to inform the calling program that CLL was finished. Then an MTR code of 15_8, requesting the control processor is made and the PP released.

NOTES:

1. The FORTRAN compiler uses CLL to load its subroutines.

2. All files loaded by CLL are compiled to execute from 0. Therefore, if a program wanted to take advantage of this feature, all K portions of the instructions must be modified for a different starting point.

3. The last overlay loaded by CLL is followed by a zero word.

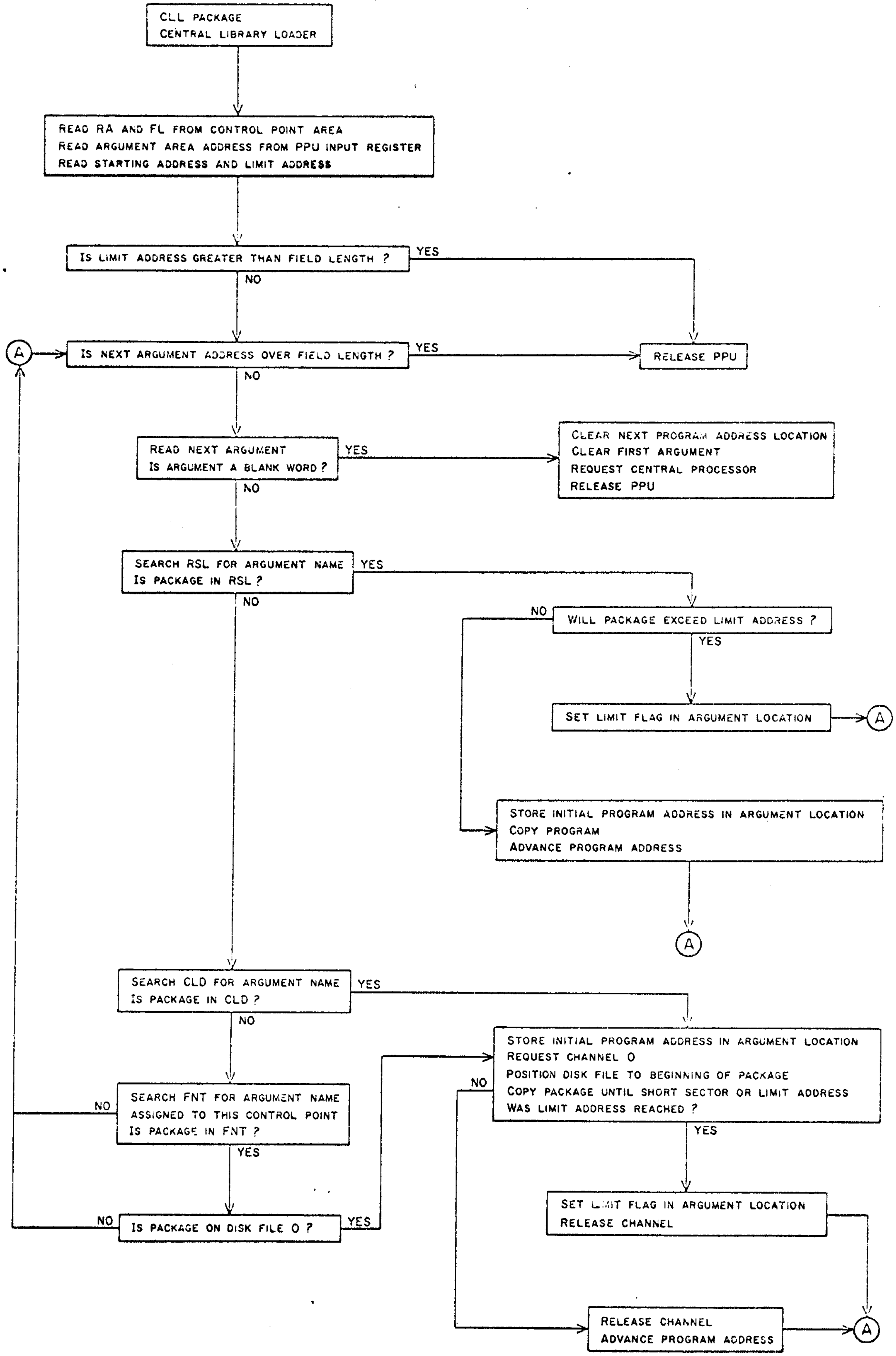
CLL Routines

1000	Main Program	1101, 1201, 15-761, 12-761,100
1100	Read arguments	12-761
1200	Process argument	1301, 1501, 1601
1300	Search RSL	
1500	Search CLD	1601
1600	Enter program from disk	741, 701, 401, 751

Direct Core Cells

1000	P10/14	CP address
	P20/24	Argument
	P50/54	Contents of Input register
	P55	RA in hundreds
	P56	FL
	P57	Constant 100
	P60/61	FWA - next available program address
	P74	CP address
	P75	Address of Input register
1100	P10/14	FWA and later LIMIT
	P50/54	Contents of Input register
	P55	RA
	P56	FL
	P60/61	Location of BA
	P62/63	FWA
	P64/65	LIMIT
1200	P01	CP assignment
	P06	Track number
	P07	Sector number
	P10/14	FNT entry
	P20/24	Contents of Input register
	P30/34	FNT status
	P74	Address of Input register

1300	P01	Number of words read
	P10/14	RSL entry
	P14	Total number of words transferred
	P20/24	Contents of Input register
	P30/34	RSL status
	P55	RA
	P57	Constant 100
	P62/63	FWA, next available program address
	P64/65	LIMIT
	P7200/7302	Input buffer
1500	P06	Track number
	P07	Sector number
	P10/14	CLD entry
	P20/24	Input register
	P30/34	CLD status
1600	P01	Sector length
	P04/05	Number of words to LIMIT
	P06	Track number
	P07	Sector number
	P20/24	Input register
	P55	RA
	P60/61	BA
	P62/63	FWA
	P64/65	LIMIT


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
NO
SEARCH FNT FOR ARGUMENT NAME
ASSIGNED TO THIS CONTROL POINT
IS PACKAGE IN FNT ?
YES
YES
SET LIMIT FLAG IN ARGUMENT LOCATION
RELEASE CHANNEL
NO
IS PACKAGE ON DISK FILE 0 ?
YES
RELEASE CHANNEL
ADVANCE PROGRAM ADDRESS
A

ROUTINE: LBC -- Loading Binary Corrections

PURPOSE: To load binary cards from the INPUT file into central memory.

GENERAL: The lower 18 bits of the input register contain a beginning address for the card loading. If the address is zero, the binary cards are loaded beginning at RA. It may be called via a control card or from a DIS console.

METHOD:

1. From the control point status word (20), the RA and FL are read.

2. Each entry in the FNT is searched for type local and assignment to this control point.

3. If no entry is found, then the PPU is released without a diagnostic.

4. When an entry is found, the file name is checked against INPUT. If it does not match, the search of FNT continues.

5. After the INPUT file is located, the FST entry is checked. The file must be on disk 0 or the PPU is released.

6. The last buffer status is checked. If it is even, then the file is being used and no action will be taken. If it is odd, then the file has no operation begin performed on it, so the status is decreased by one to make it active. When another PP wants to access this file, the buffer status will reflect an even number informing the requesting PP that the file is being acted upon.

7. The disk is positioned to the track stated in the FST and one sector is read into PP memory. After each read, a check is made for file mark and data exceeding field length. If a file mark is encountered, the buffer status is made odd and the PP released. A dayfile message "LBC RANGE LIMIT" appears if the field length would be exceeded thereby also causing the buffer status to be changed and the CP aborted.

8. After the sector read is checked, it is transferred to central memory at the location specified from the input register.

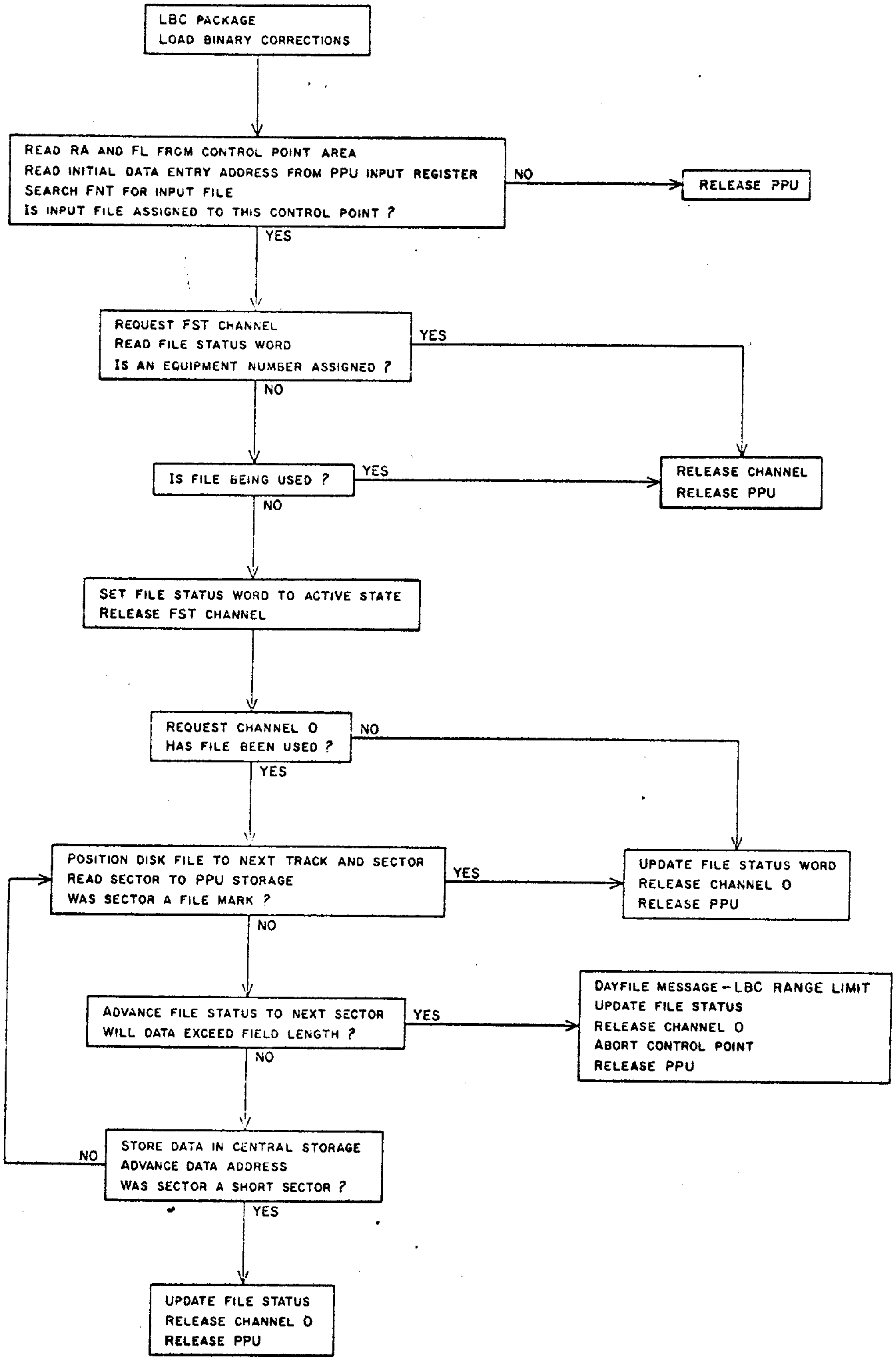
9. Only one record will be read from the INPUT file so when a short sector is encountered, the buffer status is changed and the PP is released.

LBC Routines

1000	Main Program	1200, 740, 700, 400, 750, 12-760, 530, 13-760
1200	Search for Input file	740, 12-760, 750

Direct Core Cells

1000	P01	Sector length
	P06	Track number
	P07	Sector number
	P10/14	CP status (word 20)
	P20/24	FST entry
	P50/54	Contents of input register
	P55	RA (in hundreds)
	P56	FL
	P74	CP address
	P75	Address of input register
	P7200/7702	Disk buffer
1200	P01	File type of local and CP
	P10/14	FNT entry
	P20/24	FNT status
	P20/24	FST entry
	P50/54	Contents of input register
	P57	FST address


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
NO
STORE DATA IN CENTRAL STORAGE
ADVANCE DATA ADDRESS
WAS SECTOR A SHORT SECTOR ?
YES
UPDATE FILE STATUS
RELEASE CHANNEL 0
RELEASE PPU

ROUTINE: LOC -- Load Octal Corrections

PURPOSE: To make octal corrections to a program already residing in central memory.

GENERAL: Three calls may be made to this package:

a) A call without parameters will change the central memory words specified on cards in the next INPUT record.

b) With one parameter, central memory is cleared from RA to the address specified and the cards in the next INPUT record are assembled.

c) Two parameters cause the memory between the two arguments to be cleared and then the correction cards to be read.

METHOD:

1. RA and FL are read from CP (word 20).

2. The arguments, beginning address and terminal address, of where the corrections are to be inserted are checked.

   a) First greater than second.

   b) Second greater than field length.

3. If the two arguments are not equal, the central memory contained within the two is cleared.

4. The FNT is searched for a file INPUT associated with this control point and of local or common type. If one is not found, no diagnostic results, but the PP is released.

5. The proper file must be on disk file 0 and have an odd buffer status (not busy). If either condition is not met, then the PP is released.

6. By decreasing the buffer status by one, this control point puts the file in active status.

7. Channel 0 is requested for the disk which is positioned to the proper track from the FST. The PPU buffer is filled with the octal correction cards from INPUT until the buffer is either full or a short sector is encountered.

8. The cards have trailing spaces suppressed by a zero byte and are written in 100_8 word sectors. Since the buffer is 5000_8 PP words long, many sectors may be read. Each sector has a two word control byte which is not useful data to the program. In order to have all the useful data packed, the last two words of the previous sector are temporarily stored out of the buffer and the next sector is read over their initial location. When the

control bytes have been used, the two words are restored to their buffer positions and the last two words of the sector just read are temporarily stored out of the buffer.

9. When the buffer is either filled or all the correction cards read, INPUT is put into an inactive state (status is odd) so that another PP may use it.

10. Each octal correction card is unpacked into a character string buffer (one character per word). A zero byte terminates the unpacking of one card.

11. When the line buffer is loaded, the address is assembled. The address must be between column 1 and column 7. Spaces are suppressed and leading zeroes are not necessary. If a non-octal digit appears, the address is not assembled and no diagnostic is given.

12. After the address is assembled, the data word is packed. The data must begin after column 7 and contain 20 digits. If a non-octal digit appears the word is not assembled and no diagnostic is given.

13. The assembled address is checked against field length and is not inserted into its position if it exceeds FL. The assembled word is then entered into its assembled address.

NOTES:

1. If corrections are to be made to a binary deck, LBC (load binary cards) should be used before LOC. LOC only makes changes to programs already in central memory.

2. Central memory may be cleared using LOC only if an empty record appears in the INPUT file.

3. On the correction cards, the address must end before column 7. Spacing is not important and leading zeroes may be dropped.

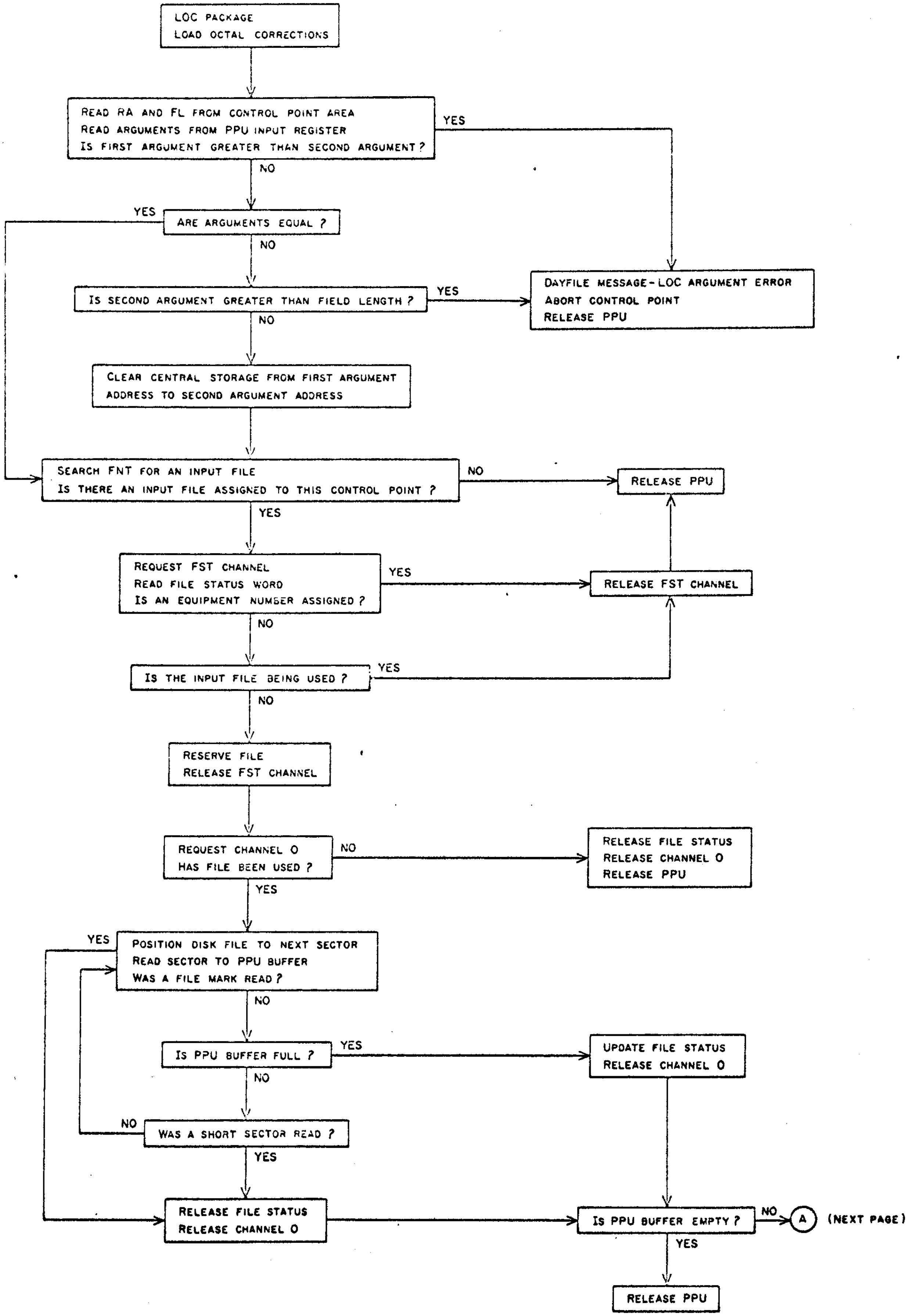
LOC Routines

1000	Main Program	1500, 1100, 1200, 1400, 1600 1300, 12-760, 100
1100	Search For Input File	12-760, 100, 740, 750
1200	Load Buffer	740, 700, 400, 750
1300	Assemble Word	
1400	Unpack Character String	
1500	Clear Storage	530, 13-760
1600	Assembled Address	

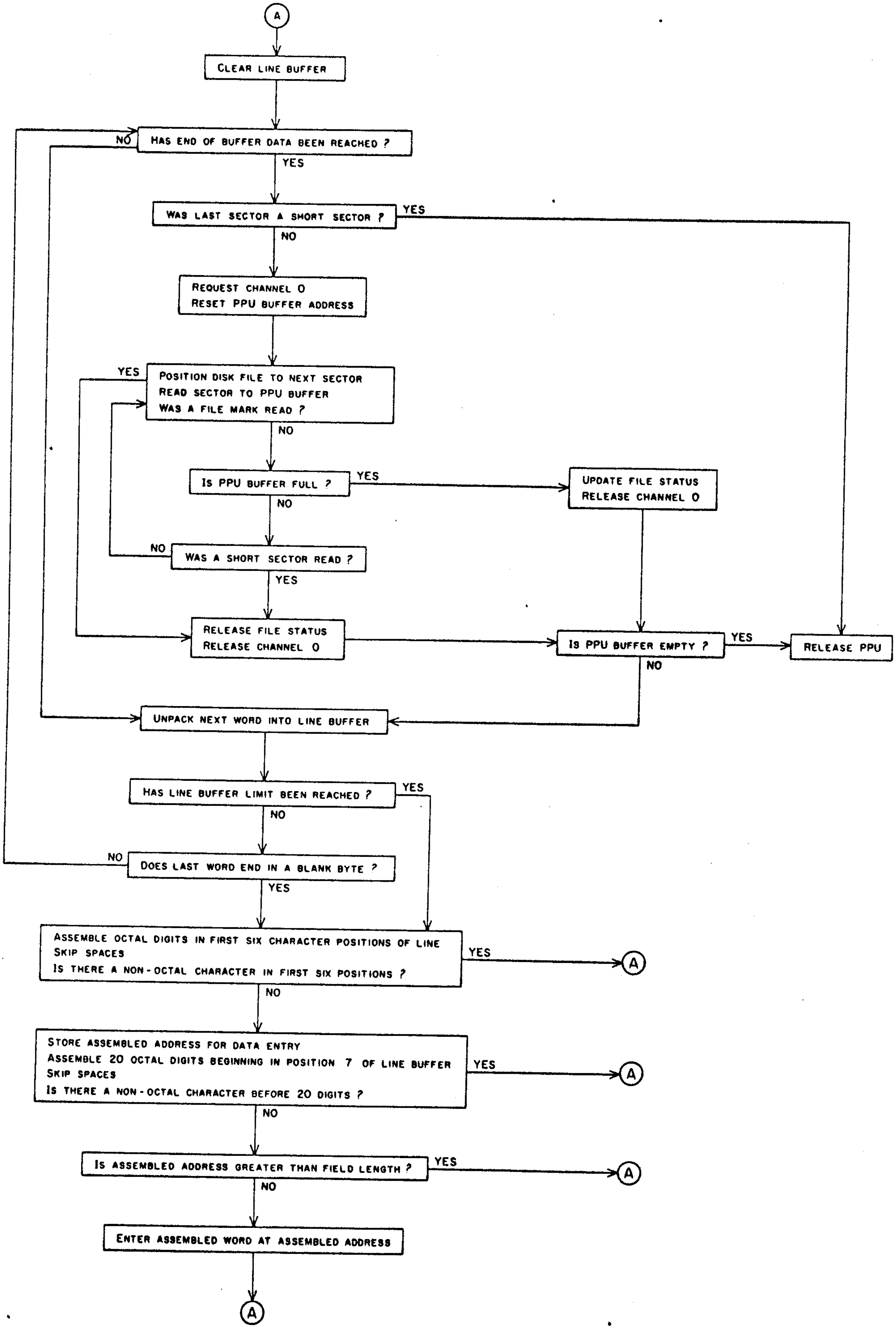
Direct Core Cells

1000	P20/24	FST entry
	P40/44	Assembled word
	P50/54	Input register
	P55	RA (in hundreds)
	P56	FL (in hundreds)
	P60	Input buffer address
	P61	Output buffer address
	P63/64	Assembled address
	P75	Input register address
1100	P01	Local type and CP assignments
	P10/14	FNT entry
	P20/24	FNT status
	P20/24	FST entry
	P50/54	Input register
	P57	FST address
1200	P01	Sector length
	P06	Track number
	P07	Sector number
	P20/24	FST entry
	P46	Data byte
	P47	Data byte
	P60	Buffer input address
	P2000/7000	Buffer

1300	P01	Octal digit
	P02	Byte address
	P40/44	Assembled word
	P62	String address
1400	P60	Input
	P61	Output
	P62	String address
	P7200/7400	String buffer
1500	P10/14	Zero word
	P50/54	Input register
	P55	RA
	P56	FL
	P62/63	First argument
1600	P01	Octal digit
	062	String address
	P63/64	Assembled address


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
HAS END OF BUFFER DATA BEEN REACHED ?
NO
YES
WAS LAST SECTOR A SHORT SECTOR ?
YES
NO
REQUEST CHANNEL 0
RESET PPU BUFFER ADDRESS
POSITION DISK FILE TO NEXT SECTOR
READ SECTOR TO PPU BUFFER
WAS A FILE MARK READ ?
YES
NO
IS PPU BUFFER FULL ?
YES
NO
UPDATE FILE STATUS
RELEASE CHANNEL 0
WAS A SHORT SECTOR READ ?
NO
YES
RELEASE FILE STATUS
RELEASE CHANNEL 0
IS PPU BUFFER EMPTY ?
YES
NO
RELEASE PPU
UNPACK NEXT WORD INTO LINE BUFFER
HAS LINE BUFFER LIMIT BEEN REACHED ?
YES
NO
DOES LAST WORD END IN A BLANK BYTE ?
NO
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

ROUTINE: MSG - Dayfile Message

PURPOSE: To enter messages from a central memory program into the dayfile.

GENERAL: This package checks the legality of the characters to be displayed and transmits them from central memory to this PP's message buffer. The lower 18 bits of the input register contains a beginning address of the message to be displayed.

METHOD:

1. The field length from CP(20) is read and the argument address of the message must be in bounds.

2. The message is checked character by character for legal display codes (0-60_8) and if all are legal, they are stored in the message buffer area of the PP.

3. A dayfile message "MESSAGE FORMAT ERROR" appears if:

   a) The argument address is not within the field length.

   b) There is an illegal character in the message.

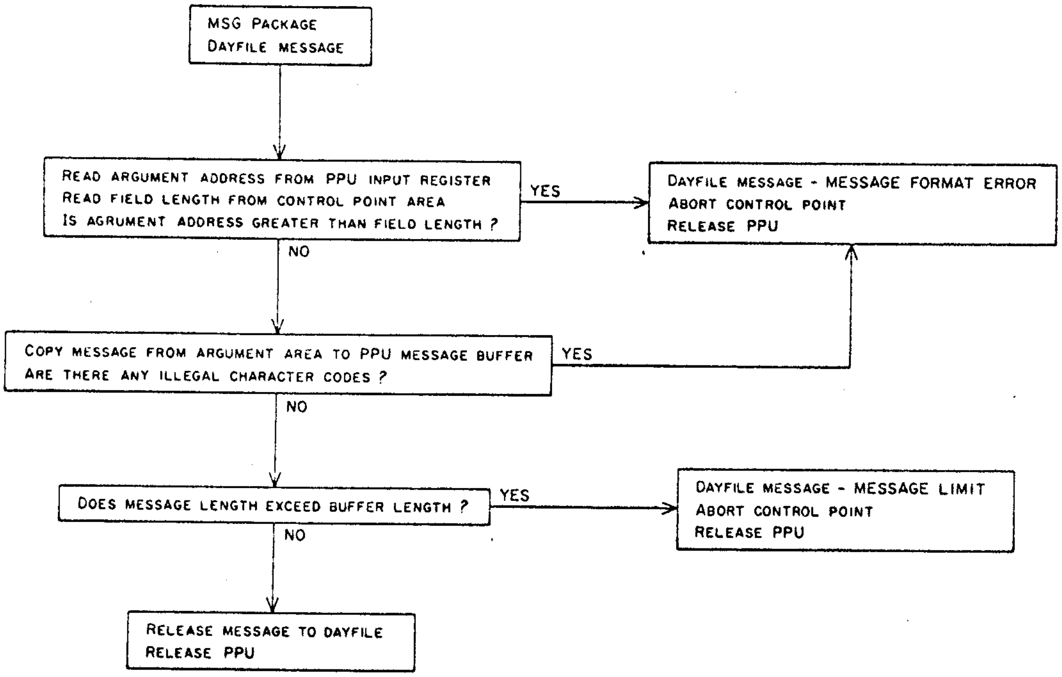
   c) The message length is greater than 6 central memory words.

4. In CP(22) there is a count of the total number of messages sent to the dayfile from the job assigned to this control point. If more than 100_8 messages have been sent, a dayfile message of "MESSAGE LIMIT" appears and CP aborted.

5. A MTR code of 01 (dayfile message) is sent to the PP resident and after it has been processed, the PP is released.

NOTE

1. A zero byte terminates the message. This byte must appear in the lowest 12 bits of the word.


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

ROUTINE: PBC - Punch Binary Cards.

PURPOSE: To format an area of central memory and punch it in the form of binary cards.

GENERAL: This package may be called by a control card or DIS console. Four calls may be issued:

1. no parameters - a binary deck beginning at RA and terminating one address less than the field length specified in the first word of the program. This call may be used to punch either a central or peripheral program in binary form.

2. one argument - area between RA and the address are punched.

3. two arguments - first argument is initial address and second is terminal address for a binary deck.

4. flagged - 400000_8 argument - initial address specified by 400000_8 + address. Lower 18 bits of this address added to it to form terminal address.

METHOD:

1. The initial address for the binary deck is read from the input register.

2. A check is made for the special 400000_8 call. If the eighteenth bit of the terminal address is set, then the lower portion of the address (that left after 400000_8 is subtracted) is set as the initial address. The lower 17 bits of this location is added to the initial address and used as the terminal address for the binary deck. Therefore, only a limited amount of memory may be punched if the 18th bit flag is set.

3. If the initial address is greater than the terminal address, the package is released without a diagnostic.

4. If the initial and terminal addresses are equal, then the lower 18 bits of RA is used as a terminal address. The initial address is cleared so that the area between RA and the FL-1 will be punched.

5. When the initial and terminal addresses have been set up properly, MTR is requested to assign the card punch to this job. If no card punch is available, the processing must wait on assignment.

6. Card punch assignment causes channel and synchronizer references within the package to be modified according to the entries from the EST.

7. Since the card punch is generally the slowest piece of equipment and PBC retains control of the PP until the complete binary deck is punched, a pause for MTR to adjust RA and FL during storage move is issued after every card is punched.

8. "PBC RANGE ERROR" and control point abort result if the terminal address ever becomes greater than the field length.

9. The punch buffer is loaded with data for the next card. In column one is stored 7-9 punches and card length. The data bytes are summed and stored in column two module 4095. Column 79 is not used and the binary sequence number is stored in column 80.

10. The punch must be ready or a console message "PUNCH NOT READY" is sent.

11. One card is then punched.

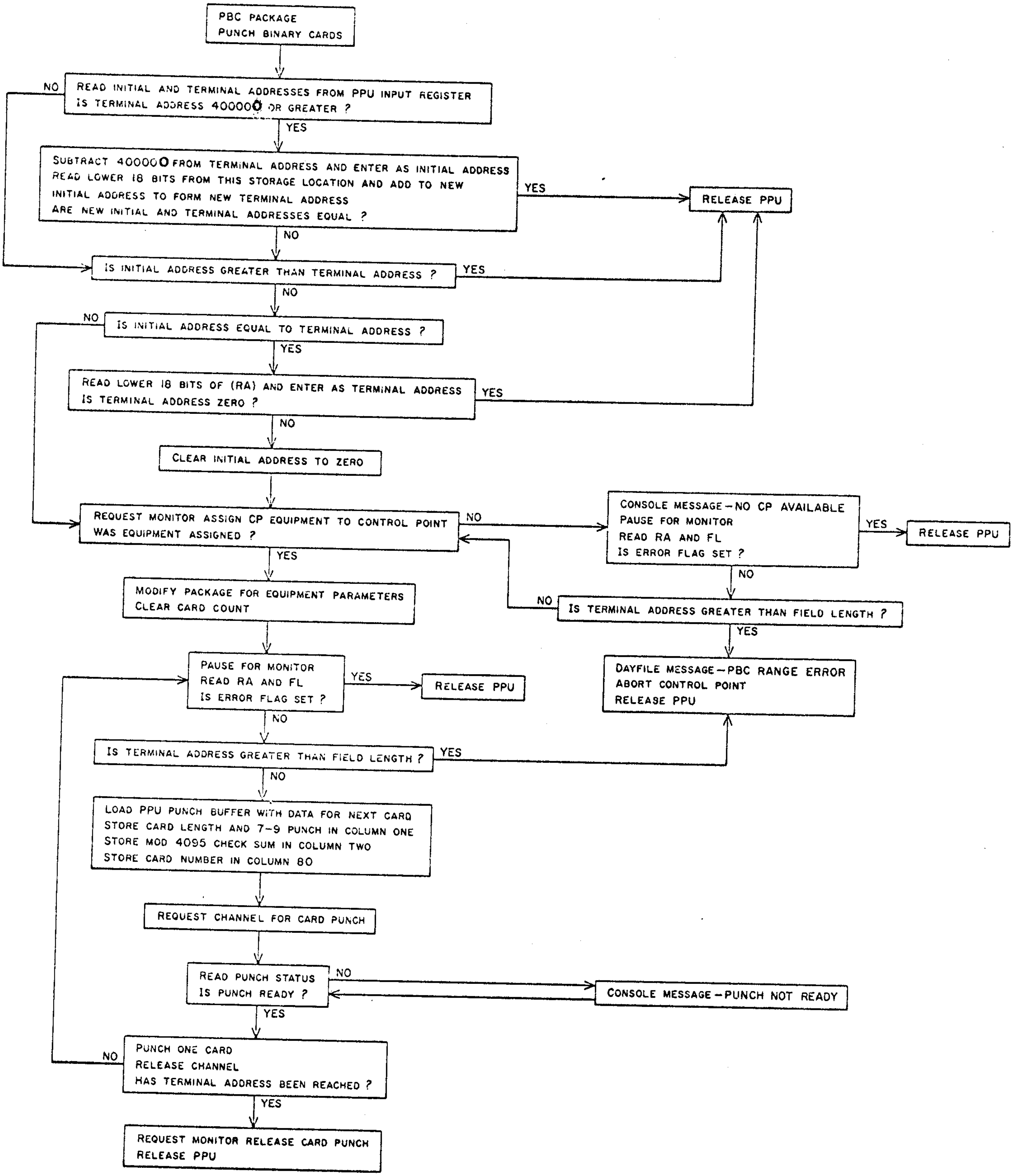
12. When the terminal address is reached, the package is released so that one less than the terminal address words are punched.

NOTES:

1. The flagged call is used by the Fortran compiler to punch a deck in I mode.

PBC Routines

1000	Main Program	1640, 1600, 1200, 1500, 1100, 1300, 1400, 23-760, 12-760, 100
1100	Sense CP status	17-760, 530, 13-760, 12-760, 100
1200	Request CP	22-760, 1100
1240	Sense punch ready	
1300	Load Punch buffer	
1400	Punch one card	13-740, 1240, 1440, 750
1440	Output one byte to punch	
1500	Modify program for equipment parameters	
1540	Channel modification table	
1560	Synchronizer modification table	
1600	Process RA length	12-760, 100
1640	Process flag length	


PBC PACKAGE
PUNCH BINARY CARDS
READ INITIAL AND TERMINAL ADDRESSES FROM PPU INPUT REGISTER
IS TERMINAL ADDRESS 400000 OR GREATER ?
NO
YES
SUBTRACT 400000 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
YES
NO
IS TERMINAL ADDRESS GREATER THAN FIELD LENGTH ?
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

Program Partitioning

I. Introduction

Chaining is a method used to execute a program which exceeds available storage or field length. The program is separated into a main program and any number of segments which may be called and executed as needed by the FORTRAN program. Both the main program and segments may contain one or more subroutines and/or functions. Overlays may be loaded (and executed) or replace the calling program by appropriate central program machine language action.

II. RUN Modes

A copy of the compiled program or segment(s) is always left on the disk. Either may be called (by name) and executed separately. Each partition (segment) including its subroutines must be separated from the main program or other partitions by a record separator. Two consecutive record separators must separate the last END statement from the first data card or file separator.

A. Chain Mode -- RUN(C,........)

Chain mode is comparable to 6 mode except that segments may be assembled following the main program. That is, no listing is produced and execution is assumed unless compile errors are encountered. The programs to be compiled must be a PROGRAM followed by one or more SEGMENT(s) each separated by a record separator.

B. Batch Mode -- RUN(B,........)

Batch mode is comparable to S mode except that any combination of one or more programs, subroutines, segments, or functions may be compiled. Also, a listing of the source language is always produced and execution is not assumed. Each program and segment is written on the disk as a file using the name specified on the PROGRAM or SEGMENT card. Therefore, execution may be initiated by a Program Call Card.

III. FORTRAN Usage

A. Definition of Segment

Each segment must begin with the statement:

SEGMENT name $(f_1, f_2, f_3, \ldots, f_n)$

where name is an alphanumeric identifier for the segment. This is the name that must be used when calling the segment

f_1, f_2, ..are file names of the files used any place in the program. These file names must agree in number and order with those specified for the main program. All files used in the execution of the main program and all segments must be specified on the PROGRAM and all SEGMENT cards.

Compilation of segments and programs differ only in the following respect:

1. Blank common is not cleared to zero by the object code in a segment.

2. Buffer space and parameters are not initialized by the object code in a segment. They are carried over from the main program in order not to destroy any input or output when calling segments.

B. Calling a Segment

A segment is called by using the FORTRAN statement:

CALL CHAIN (name)

Where CHAIN is the subroutine that loads and initializes execution of the called segment.

name is the identifier of the segment to be loaded and executed.

Segments to be called by CHAIN may reside as a named file on the disk. The only parameter to CHAIN must be the segment name.

C. General

1. Segments may be called from either the main program or another segment.
2. Calling of a segment causes the segment to be loaded over the calling program thus destroying the main program or segment that issues the call.
3. Segments may be called more than once.
4. Parameters and communication between segments can be passed only through the use of blank common.

5. Each segment is compiled beginning with relative address zero (RA = 0).
6. In order to match locations of blank common, all elements of blank common must be described in the same order and number in the main program and all segments <u>or</u> the length of common must be declared on the RUN card.

Example:

```
      CHNTST, 1, 100, 40000
      MODE 7.
      RUN (B)
      CHN.
      7-8-9

*                  PROGRAM CHN (INPUT, OUTPUT, TAPE10)
**                 COMMON  I, J, K, A(5), B(10)
                   READ 5, A
                           .
                           .
                           .
                   CALL CHAIN (S2)
                   END
      7-8-9
*                  SEGMENT S1 (INPUT, OUTPUT, TAPE 10)
**                 COMMON  I, J, K, A(5), B(10)
                           .
                           .
                           .
                   WRITE (999, 10) B(10)
                           .
                           .
                           .
                   CALL CHAIN (S3)
                   END
      7-8-9
*                  SEGMENT S2 (INPUT, OUTPUT, TAPE10)
**                 COMMON  I, J, K, A(5), B(10)
                           .
                           .
                           .
                   CALL CHAIN (S1)
                   END
      7-8-9
*                  SEGMENT S3 (INPUT, OUTPUT, TAPE10)
**                 COMMON  I, J, K, A(5), B(10)
                           .
                           .
                           .
                   END
      7-8-9
      7-8-9

                   Data Deck
    6-7-8-9
```

* These statements must specify all file names even though they are not referenced in the segment or program.

* All elements must be included in the list.

IV. Machine Language Calls

Two peripheral packages are available for loading and/or executing segments. One loads one or more segments. The other loads and executes one segment or program destroying the calling program.

A. EXU

This package loads a program to replace the calling program and initiates execution of the loaded program. The calling program is destroyed.

1. CALL

The routine is called by setting certain parameters into RA+1 of the calling program.

```
RA+1 = EXU00.........0LLLLLL

       18     24        18      bits
```

when EXU is in display code,

LLLLL is the address of the argument. The argument is the name of the central program to be loaded and executed. The name is specified in display code with trailing spaces.

2. Usage

After the monitor recognizes the request in RA+1 and assigns a PPU to process the request, RA+1 is cleared to zero by the PPU. At this point, the central program must terminate itself normally in order to allow the PPU to load the program. The central program is terminated by placing END (trailing spaces) in RA+1 and looping until it is terminated.

EXU resets or clears all operational registers - A_n, B_n, X_n - before executing the called program.

EXU loads only from job files on disk 0 (common or local)

3. Example:

Following is an ASCENT subroutine which may be called from a FORTRAN program to call EXU. This example is very similar to the CHAIN subroutine except the name of the program is fixed to SEG1.

```
Col.   2     7       11

             ASCENTF  SUBROUTINE  LDS
                  PS
                  PS
       EXIT       PS
       TAG1       SA1 = 1
```

```
                NZ  X. TAG1    .ASSURE RA+1 = 0
                SX6=053025B
                LX6 42
                SX1=SEG1
                IX6=X6+X1
                SA6=1          .SET RA+1 TO EXU PARAMETER
TAG2            SA2=1
                NZ  X2 TAG2    .WAIT FOR PPU TO ACCEPT CALL
                SX7=051604B
                LX7 42
                SA7=1          .SET RA+1 TO END
TAG4            ZR  BO BO TAG4 .WAIT FOR THE PROGRAM TO TERMINATE
         ..
SEG1            CON 23050734000000000000B
         END
```

B. CLL

This package loads one or more central programs or segments into an area of memory specified by the calling program.

1. Call

This routine is called by setting certain parameters into RA+1 of the calling program.

```
RA+1 = CLL 0.....0 BA
       18     24   18   bits
```

BA		FIRST
BA+1		LIMIT
BA+2	PROG 1	P1
BA+3	PROG 2	P2
BA+n+1	PROG n	Pn
BA+n+2	(zero)	

where CLL is in display code
      BA is an 18 bit address where the parameters are located
   FIRST is the beginning address for loading the first program.
   LIMIT is the limit address for loading the programs
   PROG1
   PROG2
   PROGn are the names (in display code with trailing spaces) of the programs or segments to be loaded.
   P1,P2
   ..,Pn are set by CLL after loading the programs and are the beginning addresses of the associated overlays.

All of the parameters except Pn must be set up by the calling program prior to setting RA+1.

2. Usage

CLL loads the programs one at a time beginning with the name specified at BA+2. The order of search for locating the overlays is:

1. Resident Subroutine Library - RSL
2. Central Library Directory - CLD
3. Assigned Job Files - common or local

The programs are loaded into the consecutive memory locations beginning with FIRST. No program may be loaded beyond the address specified by LIMIT. After a program is loaded, its beginning address is entered into the lowest 18 bits of the respective parameter word. After Cll has completed the call, BA is cleared to zero.

If program cannot be located, the address Pn for the program is not modified by CLL. If a program exceeds LIMIT, the value 777777 is entered into the respective address Pn. The last parameter must be followed by a full word containing zero.

It should be remembered that programs and segments compile with a reference address beginning with zero (000000). Since the central program calling CLL resides at zero, the loaded programs (by CLL) will not have proper address terms for those instructions containing 18 bit address. Therefore, the user must modify the addresses of the loaded program or use some addressing scheme where the calling program defines a pseudo-reference address in an index register whenever memory is referenced.

CONTROL DATA CORPORATION

Development Division - Applications

SYSTEM PERIPHERAL PACKAGES AND OVERLAYS

Chippewa Operating System

10/21/65

Table of Contents

SYSTEM PERIPHERAL PACKAGES AND OVERLAYS

INTRODUCTION

All peripheral packages that begin with a numberal are special operating system packages or equipment driver overlays. The system packages begin with the numeral "1" and begin execution at address 1000 of peripheral memory. Their functions are to load jobs onto the disk, make control point assignments, process the control statements, and print the jobs' output. Whenever specialized operations , i.e. read tape, punch cards, translate control statements, etc., are required, an overlay is loaded into the requesting PP at location 2000. These overlays begin with the numberal "2" and parameters are passed to them by direct core cells ($1\text{-}74_8$). Most of them are maintained in RPL (resident peripheral library), however they could be kept in PLD (peripheral library directory) if the system packages searched this table. Since most of them are fairly short, the system packages expect them to reside in central memory.

ROUTINE: 1AJ - Advance Job.

PURPOSE: To advance the status of a job by controlling the processing of the next control card or terminating the job.

GENERAL: This package is called by MTR on its main loop and the following conditions prevail when 1AJ is called.

a. A job has been assigned to a control point by 1BJ.

b. The central processor is not executing the job at the control point.

c. The storage move flag is not set.

d. The control point is not listed in the CPU stack, i.e., it is not waiting on the central processor.

METHOD:

1. If an error flag for the control point is set, 2EF is called to process the error. This routine will issue the proper error diagnostic to the dayfile and then position the control card buffer parameters to the statement after an EXIT card or, if no EXIT card to found, to the record separator.

2. 2TS is called to process the control statements in the order encountered and all the statements will be processed before 1AJ regains control.

3. If the control point has zero priority, i.e., PP program that uses central memory, all files and equipment assigned to this control point are dropped by 2DF and monitor. A request is also made to monitor to release the storage reserved by this control point and a pause loop is maintained until the field length is zero. The control point is then cleared of information and 1AJ is released. No dayfile data will be written in this case.

4. In the normal case with a priority set at the control point, an attempt is made to locate an "OUTPUT" circular buffer so that it may be emptied if it is not. The first 100_8 words of the program are searched for the buffer. The lower 18 bits of each of these words specify an address where the file name and status is located. If the address is within the field length, the name is checked for "OUTPUT." The search continues until RA+100_8 words have been checked.

5. If the buffer status indicates that a file mark has already been requested, it is assumed that the buffer is emptied of usable information. If the file mark is not set, then the buffer will be dumped if

   a. it is a disk file

   b. the last operation was a write.

   2WD is called to write the buffer contents on the disk.

6. Both the amount of central processor and peripheral processor running time is read from the control point, converted to decimal, and sent to the dayfile.

7. A search is made of FNT to find a file named "OUTPUT" assigned to this control point. If there is none, then such a name is entered into the FNT so that the dayfile can be printed.

8. The file name is then changed to that of the job name and the job's priority is also put into the FNT. The file is released from the control point by putting a zero value in the control point byte. This action will cause the print routines (1DJ or 1TD) to sense a file ready for printing.

9. All files assigned to this control point in the FNT are dropped by 2DF. The FNT/FST entries are completely zeroed.

10. The upper most byte of the EST has the control point assignment for the equipment. All the pieces of equipment assigned by this job are released by a monitor request.

11. The control point area is then cleared and 1BJ is called to this PP so that another job may be assigned.

1AJ ROUTINES

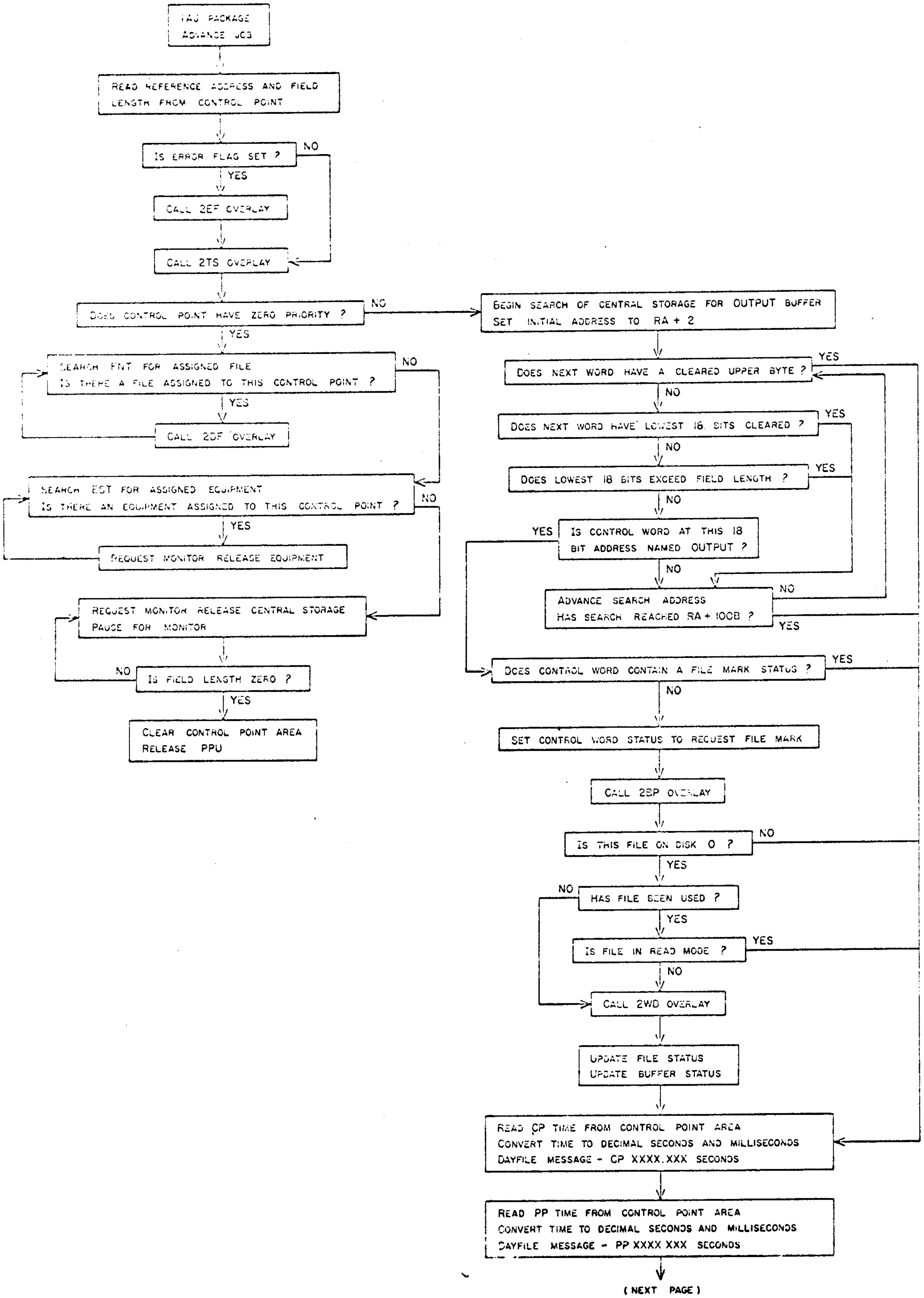
1000	MAIN PROGRAM	1700, 1410, 1740, 1500 1100, 1320, 100, 12-760
1100	RECORD RUNNING TIMES	1200, 530, 530
1200	DECIMAL CONVERSION	
1320	RELEASE OUTPUT FILE	1640
1410	DROP FILES	1700, 23-760
1500	SEARCH FOR OUTPUT BUFFER	1700
1640	BEGIN OUTPUT FILE	740, 750
1700	CALL SUBROUTINE	2000
1740	CLEAR CP AREA	10-760, 17-760

DIRECT CORE CELLS

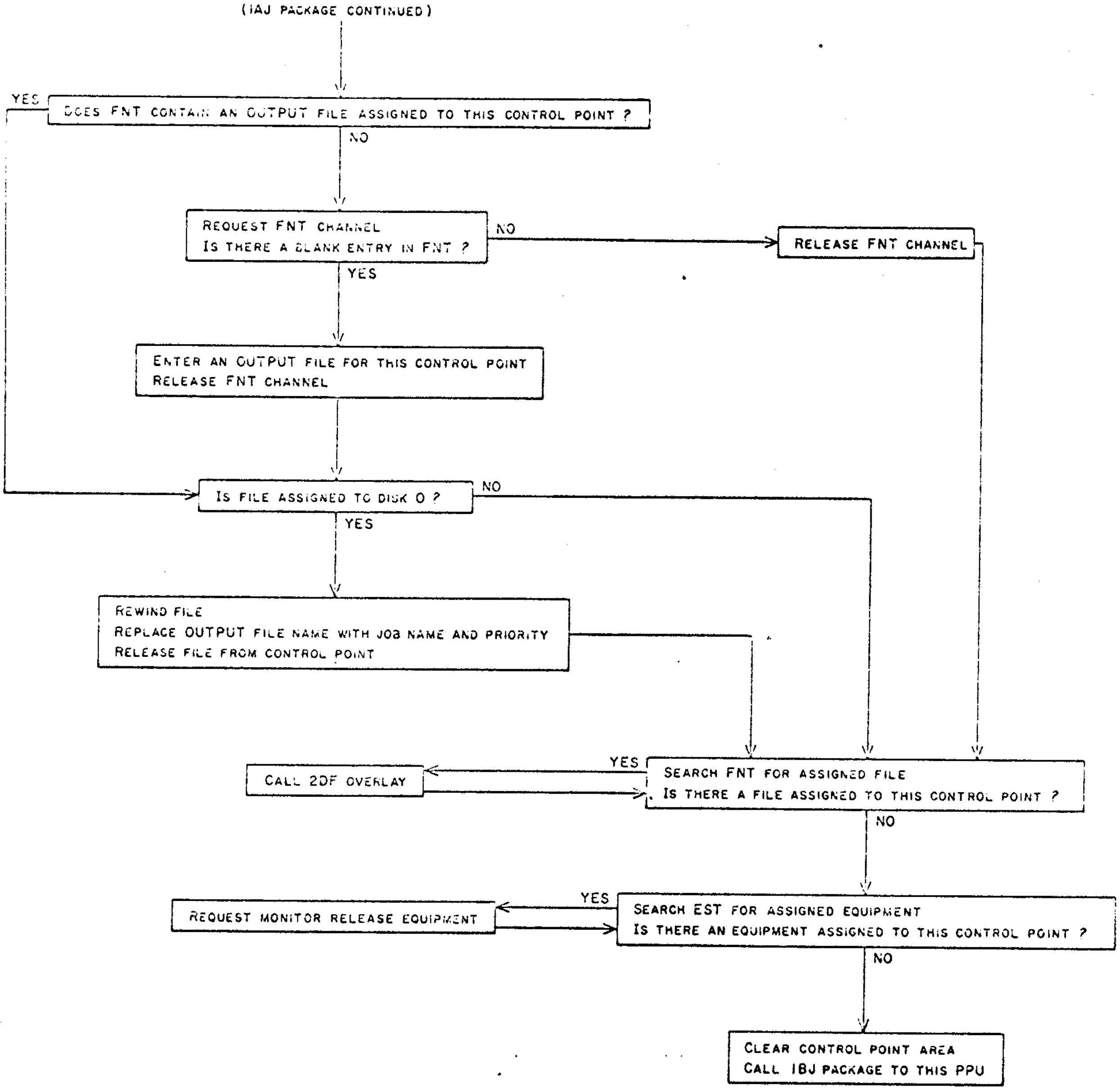
1000	P10/14	CP STATUS
	P55	RA
	P56	FL
	P50/54	CONTENTS OF INPUT REGISTER
	P70	CONSTANT 1
	P71	CONSTANT 100
	P72	CONSTANT 1800
	P74	CP ADDRESS
	P75	ADDRESS OF INPUT REGISTER
1100	P01	MESSAGE WORD COUNT
	P10/14	CPTIME, LATER PP TIME
	P20/30	CP TIME MESSAGE, LATER PP TIME MESSAGE
	P74	CP ADDRESS
1200	P10/14	CP OR PP TIME

	P20/30	CP TIME MESSAGE. LATER PP TIME MESSAGE
	P71	CONSTANT 100
1320	P10/14	FNT ENTRY
	P20/24	FNT STATUS
	P40/44	CP(21) WITH ADDED PRIORITY
	P50/54	INPUT REGISTER
	P74	CP ADDRESS
1410	P10/14	FNT STATUS, LATER EST ENTRY
	P20/24	EST STATUS
	P40/44	FNT ENTRY
	P46	FIRST OF FNT
	P47	IN OF FNT
	P50/54	INPUT REGISTER
	P74	CP ADDRESS
1500	P01	SEARCH ADDRESS
	P10/14	ARGUMENTS LOCATED AFTER RA+2
	P20/24	CONTROL WORD OF ARGUMENT AT RA+2+n
	P40/44	BUFFER STATUS
	P45	LAST BYTE OF CONTROL WORD
	P54	RA
	P55	FL
	P57	FST ADDRESS
1640	P01	CONSTANT 2
	P10/14	FNT ENTRY
	P20/24	FNT STATUS
1700	(A)	SUBROUTINE NAME
	P01	RPL INDEX
	P02/03	SUBROUTINE NAME

	P04	RPL STARTING ADDRESS
	P10/14	RPL ENTRY
1740	P01	MAXIMUM 200_8 WORD COUNTER
	P10/14	CP STATUS, LATER ZERO WORD


1AJ 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
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 O ?
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)
DOES FNT CONTAIN AN OUTPUT FILE ASSIGNED TO THIS CONTROL POINT ?
YES
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
CALL 2DF OVERLAY
YES
SEARCH FNT FOR ASSIGNED FILE
IS THERE A FILE ASSIGNED TO THIS CONTROL POINT ?
NO
REQUEST MONITOR RELEASE EQUIPMENT
YES
SEARCH EST FOR ASSIGNED EQUIPMENT
IS THERE AN EQUIPMENT ASSIGNED TO THIS CONTROL POINT ?
NO
CLEAR CONTROL POINT AREA
CALL 1BJ PACKAGE TO THIS PPU

ROUTINE: 1BJ -- Begin Job

PURPOSE: To assign a job to a control point and process the job card.

GENERAL: The package is called by DSD where "X.NEXT" is requested and recalled by 1AJ. The control point assignemtn is specified in the input register upon entry to the package.

METHOD:

1. If the error flag in CP(20) is set, the package is released. The error will be processed later by 1AJ and 2EF.

2. If the priority is not zero, this is a recall entry. Otherwise, the following steps occur:

   a. A search is made through FNT for the highest priority file of TYPE 0 (INPUT) and no control point assignment. If none is found, the job name is set to NEXT (for display) and the message IDLE sent to display and the package released in recall status.

   b. If a file was found, the file name in FNT is set as job name, the file name is changed to INPUT, and the TYPE is changed to local. Also, the priority of the job is set in the CP.

3. If the job cards have been loaded, this is a recall entry. Otherwise, the following steps occur:

   a. 300$_8$ words of central memory are requested of MTR. If not assigned, the message WAITING FOR STORAGE is sent to display and the package released in recall status.

   b. If 300 words were assigned, the first ten words are set as follows:

RA = RA+1 = RA+2 =	0		
RA+3 =	INPUT	10	File Name and Buffer Status
RA+4 = RA+5 = RA+6 =	0	010	FIRST, IN, OUT
RA+7 =	0	0300	LIMIT

   c. 2BP is called to verify the parameters and set direct core cells for 2RD.

   d. 2RD is called to read the control statement record into CM.

   e. FST is updated to reflect a completed read. The control statements are moved from the CM buffer to CP control statement buffer using PP locations beginning at 7000 as a transient buffer.

f. CP(21) is set to reflect the reading of the control statements.

g. 2TJ is called to translate and process the job card. The time limit specified on the JOB card is set by MTR.

4. The field length specified on the job card is requested of MTR. If not assigned, the message "WAITING FOR STORAGE" is sent to display and the package released in recall status.

5. If MTR assigned the memory, the job card is issued to the dayfile.

6. Finally, the package is released. The remaining statements will be processed later by 1AJ and 2TS.

NOTES: All console messages are sent to display by entering the message in CP (30-37). These messages are line 3 of the control point display.

The job card is sent to the dayfile by storing it in the message buffer (address specified by P77) and issuing a F01 request to MTR.

All overlays called by 1BJ must be in RPL since PLD is not searched when calling the overlays. These overlays include 2BP, 2RD, 2TJ.

Two recall flags are used:

a. priority given by CP(22).

b. control cards loaded or not loaded by CP(21).

Three conditions may exist which will cause 1BJ to be released in a recall status. These are:

1. If there exists no unassigned input files in FNT of the TYPE input.

2. If MTR will not assign storage for the buffer to load the control cards into CM.

3. If MTR will not assign storage for the job as specified by FL on the JOB card.

Upon entry to 1BJ, two flags (see above) specify whether this is the initial entry or a recall entry. If it is a recall entry, the flags given above cause the package to skip the areas of code it has executed on a previous call. For example, if the priority given in CP(22) is zero, this is either the initial entry or no unassigned input file was found on the previous entry (same as initial entry). If the priority is not zero, a file has been assigned and the coding to find and assign a file is bypassed. If the job cards have not been loaded (specified by CP (21 byte 11-0) = 0) they must be loaded into the CP control statement buffer. If they have been loaded, this coding is skipped. If the priority is non-zero and the job cards are loaded, or after these have been done, storage is requested for the job. If not assigned, the package is released in recall status again. Upon next entry all coding will be skipped except this storage request since the priority will be non-zero and the job cards are loaded.

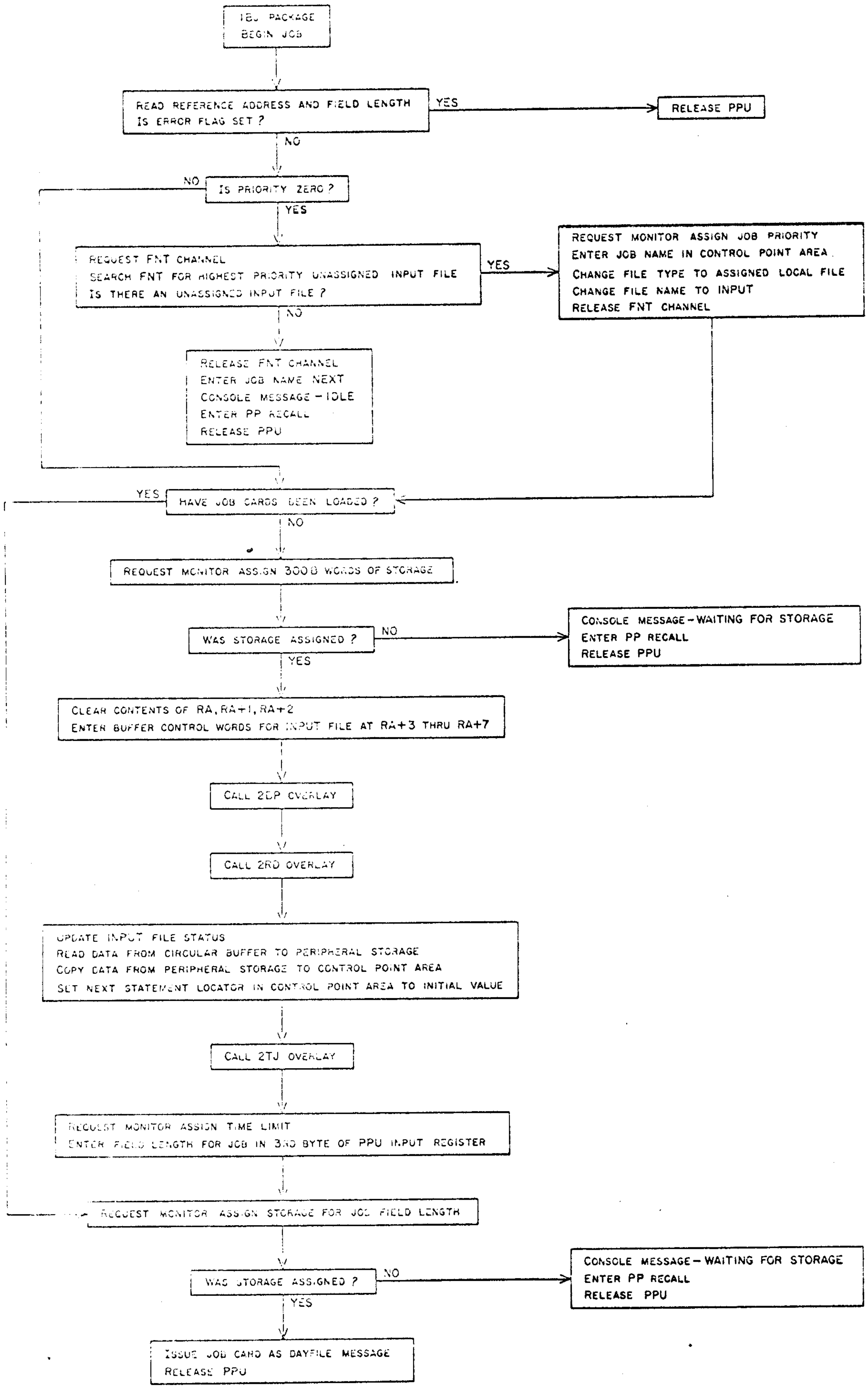
Releasing a PP in recall status involves storing the contents of the input register in CP(25) and then releasing the PP via MTR request 12. A normal release leaves CP(25)=0.

1BJ Routines

1000	Main Program	1100, 1440, 1400, 1500, 12-760
1100	Search for Job	740, 750, 12-760, 24-760
1240	Call (Overlay) Subroutine	
1300	Read Control Cards	1240
1400	Request Storage	10-760, 12-760
1440	Read Job Cards	1400, 1300, 1240, 14-760
1500	Issue Statement	01-760

1BJ Routine Direct Core Parameters

1000	P75	Address of Input register	
1100	P50/54	Contents of input register	
	P74	Address of control point	
1240	(A)	Name of overlay to be loaded and executed	
1300	P54	Field Length (FL) from CP-20	
	P55	Reference Address (RA)	
	P57	Address of INPUT FST entry.	
	P63	Lower 12 bits of IN =	after control cards are read
	P65	Lower 12 bits of OUT =	
	P70	0001 (constant)	
	P74	Address of Control point	
1400	(A)	Field length (in hundreds) needed	
	P56	Field length (FL) from CP-20.	
	P74	Address of control point	
1440	P36	Time Limit (TL) from JOB card (in tens)	
	P37	Field Length (FL) from JOB card (in hundreds)	
	P55	Reference Address (RA)	
1500	P74	Address of control point	


1BJ 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 2DP 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

ROUTINE: 1DJ - Phase 3 print

PURPOSE: To monitor the processing of an OUTPUT file.

GENERAL: DSD calls 1DJ to a control point to print a jobs' output. The package appears as "PRINT" and is loaded at dead start when "AUTO" is typed or whenever "X.PRINT" is typed. It remains in recall state and is available to print an OUTPUT file when one is released.

METHOD:

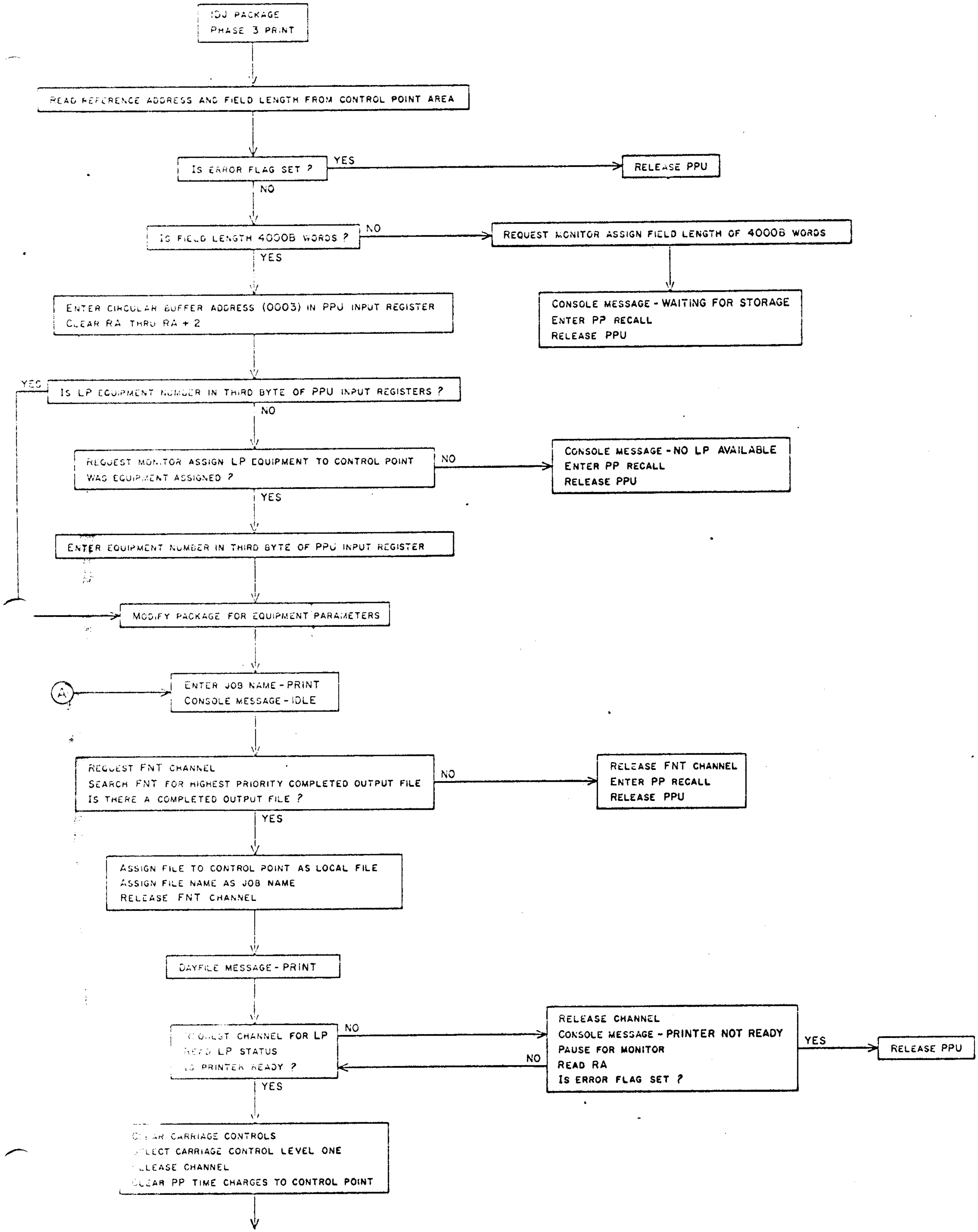
1. 4000_8 words are requested from MTR. When memory has been allocated, a line printer is requested and the package is modified for the equipment parameters.

2. The FNT is searched for an "OUTPUT" file and the message "IDLE" is displayed until such a file is found. When found, a "PRINT" entry is made in the dayfile and when the printer becomes ready, the file name is changed to the job name in FNT. At the control point the job name appears instead of "PRINT" and the console message is changed from "IDLE" to "PRINT".

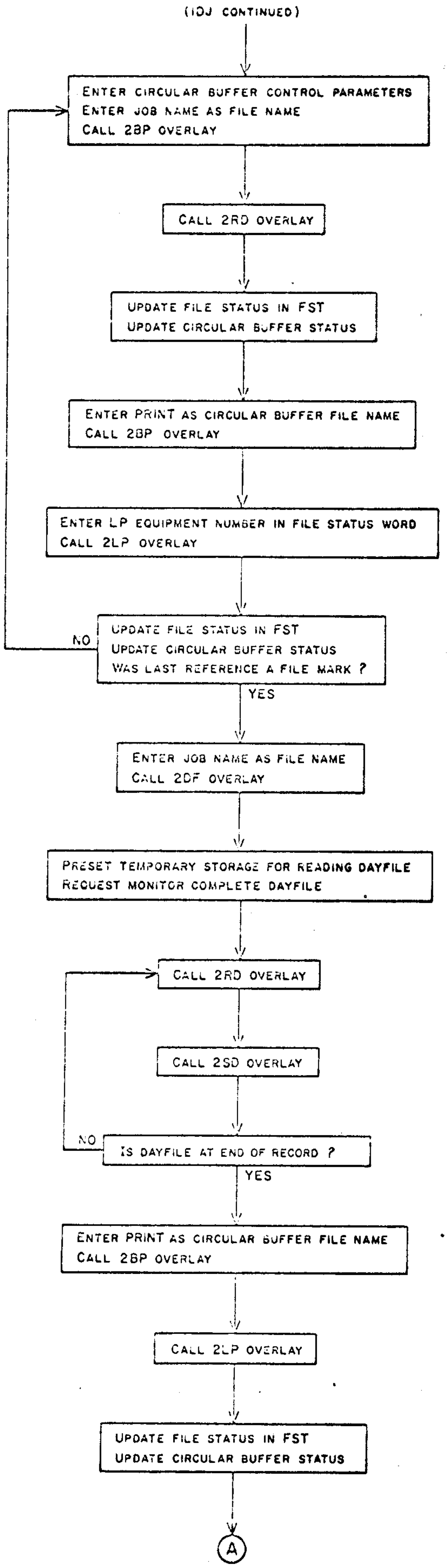
3. 2RD is called to read from the disk to the circular buffer in central memory. The reading continues until the end of the file is encountered or until the central memory buffer will not hold another full sector.

4. 2LP is then called to print this information and will continue printing until there is no more data in the buffer to print.

5. If an end-of-file has not yet been detected, control continues at step 3. When it is detected, the dayfile is searched for entries belonging to this job and then the entries are printed. Control reverts back to step 2.


IOJ 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
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
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)


(IOJ 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

ROUTINE: 1LJ -- Phase One Card Load

PURPOSE: To build up an input file from the card reader onto the disk.

GENERAL: 1LJ is the "READ" package which is called in by DSD when "AUTO" is typed at dead start. When "READ" is assigned a control point, it remains in recall state and is available to read a job whenever the card reader becomes ready.

METHOD:

1. The job name READ is stored in CP(21). The error flag is checked and if an error is sensed, the PP is released. READ must be reassigned when it is needed again.

2. If 4000_8 words (FL) have not been assigned, the routine requests the storage and puts itself into PP recall.

3. A circular buffer address (0003) is entered into the PP input register and the first 3 words (RA—RA+2) are cleared. Any central program must have 3 words reserved for system communication so that means the circular buffer parameters are located at RA+3.

4. FIRST = IN = OUT = 10_8 are the preset buffer parameters and LIMIT = 4000_8.

5. Upon entry the third byte of the input register may contain the equipment number of the card reader. If it does not, then MTR is asked for the assignment. The number will come back in the first byte of the message buffer and then is transferred to the third byte of the input register.

6. If the assignment was not completed, "NO CR AVAILABLE" is stored in CP(30) and the PP put into recall.

7. The above 6 steps are initialization procedures and are not repeated unless "READ" is dropped and must be reassigned.

8. "READ" appears as the job name in the CP and "IDLE" as a console message when no reading is being done.

9. The channel from the card reader entry in the equipment status table (EST) is requested and then the status of the card reader is checked. If the reader is not ready, the PP is put into recall and released.

10. After the card reader is found to be ready, the file name "READ" and a buffer status of 10_8 meaning requested coded read is entered into BA.

11. 2BP is called to check the legality of the buffer parameters.

12. The equipment number of the card reader from the input register is stored in the FST entry.

13. 2RC is then called to read one card.

14. The FST entry is updated and stored as is the buffer status word (BA). Both reflect an 11_8 condition completed coded read.

15. 2TJ is called to translate the job card. The job name is entered in PP(30) from 2TJ and is transferred to CP(21). Therefore, the control point assigned to READ has a new job name (from job card) and a console message of "READ" instead of "IDLE". A dayfile entry of the job name and READ is made.

16. Next READ in BA is replaced with the job name and the buffer status is changed to request coded write (14_8).

17. Again 2BP is called to verify the buffer parameters. Every write operation on the disk is terminated with an EOF record so that if a file mark was requested it is not completed so that two file marks will not be written.

18. 2WD is called to write the contents of the buffer of the disk.

19. Upon reentry to 1LJ, the FST entry and the buffer status (BA) is updated to reflect a completed coded write.

20. The file name READ and buffer status of 10_8 - requested coded read - is again entered into BA.

21. 2BP is called to determine the legality of the buffer parameters and the card reader equipment number is placed in the third byte of the input register for 2RC.

22. After 2RC returns control to 1LJ, the FST and buffer status are updated to reflect a completed coded read.

23. If a file mark was not read, then the job has not been completely read in. The contents of the buffer are written on the disk and more cards read until a 6-7-8-9 card is found.

24. When a file separator card is sensed, an MTR request (04) to update the PP running time at the control point for the requesting processor is issued. The time is converted to decimal and sent to the dayfile in the form PPXXXX sec.

25. In order to release the job to the system the job name is stored in BA and 2BP is called for a final check of the buffer parameters. The disk file is rewound by setting the current track to the beginning track in the

FST. Also the current sector byte is cleared and the last buffer status is set to 01. The priority is added to the FNT entry and the control point assignment byte is cleared. Therefore, the input file is released and ready for MTR to assign it a control point for execution.

26. "READ" with a 10_8 request is again entered into BA of the circular buffer and 2BP is called to check the parameters. An FST entry is cleared in preparation for a new file and a check is made for a ready card reader.

27. If a card reader is not ready, the PP is put into recall so that it will be able to detect when the card reader becomes ready.

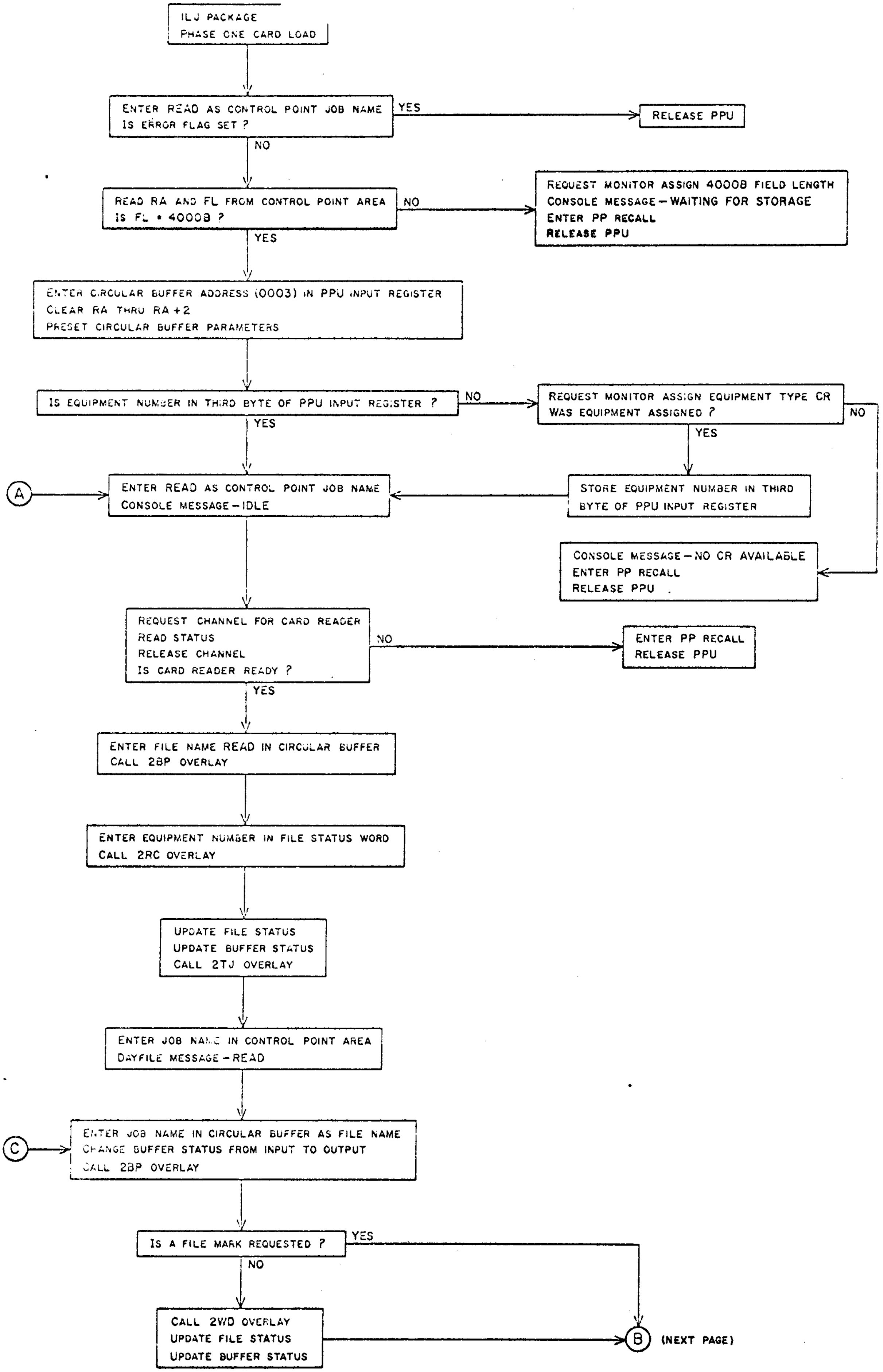
1LJ Routines

1000	Main Program	1500, 1440, 1600, 1040
1040	Process Job	1700, 1400, 530, 1100, 1300, 1200
1100	Dump Buffer	1400
1200	Release Job	1400
1300	Record Time	4-760, 530
1400	Call RPL Package	2000
1440	Request CR	22-760, 12-760, 100
1500	Enter CP Status	10-760, 12-760, 100, 1740
1600	Sense CR Ready	740, 750, 12-760, 100
1700	Load Buffer	1400
1740	Preset Buffer Parameters	

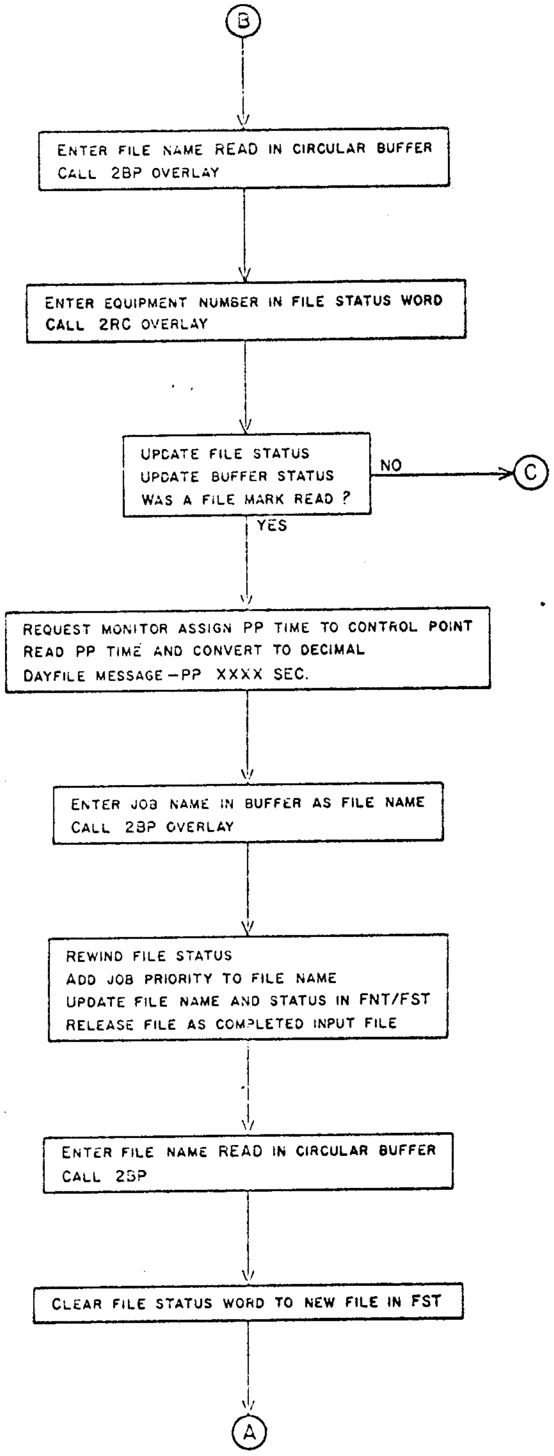
Direct Core Cells

1000	P50-54	Input register
	P70	Constant 1
	P71	Constant 180
	P72	Constant 1000
	P75	Input register address
1040	P10/14	Zero word
	P20/24	FST entry
	P74	CP address
1100	P10/14	CP(21)
	P20/24	FST entry
	P40/44	File control word (BA)
	P50/54	Input register
	P55	RA
	P57	FST address
1200	P10/14	CP(21), zero word
	P20/24	FST entry
	P35	Job priority
	P55	RA
	P57	FST address
	P74	CP address

1300	P10/14	PP time - CP(24)
	P74	CP address
1400	(A)	Package name
	P01	RPL ordinal
	P02/03	Package name
	P10/14	RPL entry
1440	P01	Constant 2
	P10/14	Message buffer
	P50/54	Input register
	P74	CP address
	P77	Message buffer address
1500	P01	Constant 3
	P10/14	CP(20), zero word
	P54	Constant 3
	P50/54	Input register
	P55	RA
	P56	FL
	P74	CP address
1600	P01	CR status
	P10/14	EST status
	P20/24	EST entry
	P50/54	Input register
	P74	CP address
1700	P20/24	FST entry
	P40/44	File control word (BA)
	P50/54	Input register
	P55	RA
	P57	FST address
1740	P10/14	Zero
	P14	10_8
	P54	Constant 3
	P55	RA
	P56	FL


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)


B
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
WAS A FILE MARK READ ?
NO
C
YES
REQUEST MONITOR ASSIGN PP TIME TO CONTROL POINT
READ PP TIME AND CONVERT TO DECIMAL
DAYFILE MESSAGE—PP XXXX SEC.
ENTER JOB NAME IN BUFFER AS FILE NAME
CALL 2BP OVERLAY
REWIND FILE STATUS
ADD JOB PRIORITY TO FILE NAME
UPDATE FILE NAME AND STATUS IN FNT/FST
RELEASE FILE AS COMPLETED INPUT FILE
ENTER FILE NAME READ IN CIRCULAR BUFFER
CALL 2BP
CLEAR FILE STATUS WORD TO NEW FILE IN FST
A

ROUTINE: 1LT Phase One Tape Load

PURPOSE: To load jobs from a magnetic tape onto the disk until an empty file is encountered.

GENERAL: The package is called by DSD after the operator types "X.LOAD." at the console. The control point for the package is specified in the input register.

METHOD:

1. Initialization of the routine involves the following steps:

   a. If the requested control point has a job name, the package is released.

   b. Otherwise, the job name LOAD is set in the CP(21) for display purposes.

   c. 10000_8 words of central memory are requested to be used as a buffer for reading tape and writing disk.

   d. If MTR does not assign 10000_8 words, the package is released.

   e. Otherwise, the CM buffer is set up as follows:

```
       RA  =  RA+1 = RA+2 = | 0             |
                     RA+3 = | 0             | File Name and Buffer Status
RA+4 = RA+5 = RA+6 = | 00          04| FIRST, IN, OUT
                     RA+7 = | 00      010000| LIMIT
```

   f. The buffer address, 0003, is stored in the PPU input register (internal) for future reference by the package.

   g. A tape assignment is requested of the operator by storing REQUEST TAPE in CP(30-37).

   h. A function 17 request is sent to MTR while waiting for the operator to assign the tape. This function is repeated until the tape is assigned. The equipment number specified by the operator is contained in CP(22).

   i. The equipment number is stored in PPU input register (internal) for future reference by the package.

2. The following steps occur for the initialization of each file (job):

   a. RA+3 is (re) set as follows:

TAPE......10 in order to read the tape files.

b. 2BP is called to verify the buffer parameters and to set up direct core parameters for 2RT.

c. 2RT is called to read information from the tape and store it in buffer in central memory.

d. The file status (LBS field) in FST is updated (odd value) to reflect the record(s) just read.

e. The buffer status (at RA) is updated (odd value) to reflect the record(s) just read.

f. If a file mark was read at this point, it would have been the second consecutive file mark and, therefore, the package (1LT) is released.

g. Otherwise, 2TJ is called to set up the job name and priority in direct core cells.

h. The job name is in the CP for display and dayfile accounting purposes.

i. The message LOAD is sent to the dayfile.

3. The following steps occur as a loop for loading the tape records onto the disk:

   a. The job name (from CP) is stored as file name before writing disk so that FNT contains the job name of type input.

   b. 2BP is called to set up direct core parameters for 2WD, i.e., also assigns the new file.

   c. 2WD is called to write the bufferin central memory onto the disk, if a file mark was not requested. The file marks are automatically handled by 2WD on every write.

   d. Again, the FST word and the buffer status are updated to reflect the record(s) just written.

   e. The buffer is again loaded as specified before in steps 2) a., b., c., d., e.

4. When a file mark is encountered on the tape (and the record(s) are written on disk), the following steps are performed to release the disk file (job) just written.

   a. The job name is stored as the file name in order to call 2BP to set up direct core parameters for rewinding the file.

   b. The file (on disk) is rewound by making the following changes to FST.

i. setting current track=beginning track

ii. setting current sector=0

iii. setting last buffer status=0001

c. The priority, from the job card, is entered into FNT.

d. The file type is set to input.

e. The file status is cleared from file TAPE by a call to 2BP and resetting FST.

NOTES: PPU time used to load the jobs on the disk is not charged to the individual jobs.

The package 1LT is released without completing the tape to disk operation if any of the following conditions arise:

1. too many control cards in a job.

2. illegal parameters on the job card.

3. no tracks are available on disk.

4. the track limit (512 tracks) is exceeded for a job.

5. the operator drops the CP.

When the package (1LT) is released, either normally or prematurely, the files (tape and disk-FNT/FST), equipment (EST), and storage (CP(20)) are released by a special section of 1AJ. This section releases these items for control points not using the CPU but using CM for buffers. 1AJ detects this when a CP has a zero (0) priority. 1AJ is entered to release the package by the master loop in MTR.

The dayfile message LOAD is written via MTR function 01 and resident routine located at 530_8.

Since the package is immediately released if 10000_8 words are not available from MTR, the operator should call LOAD after dead start. Otherwise, he will have to wait for the CP's to be relatively inactive in order not to run into any storage conflicts.

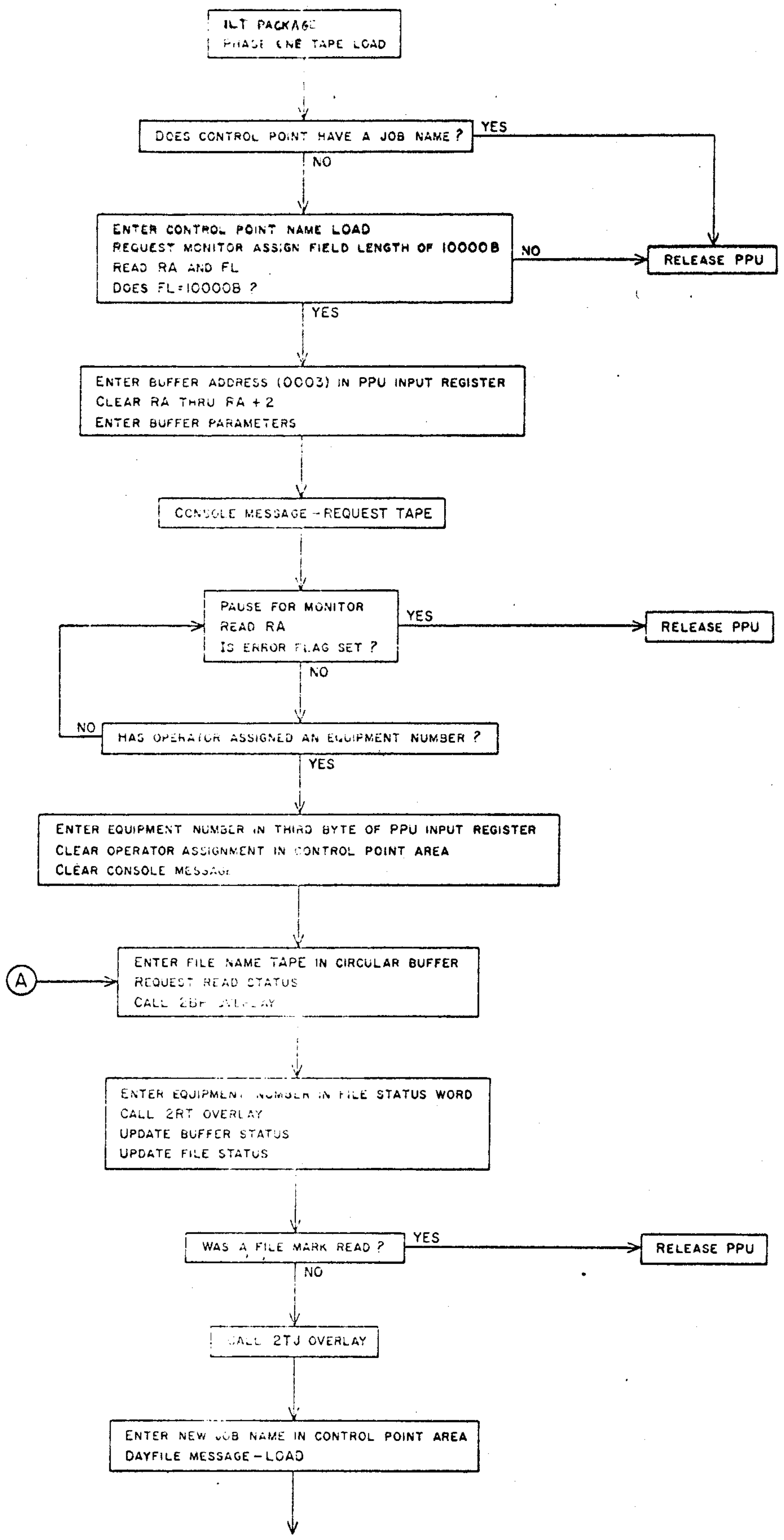
All overlays called by 1LT must be in RPL since PLD is <u>not</u> searched when calling the overlays. These overlays include 2BP, 2RT, 2WD, 2TJ.

1LT Load Tape Routines

1000	Main Program	1300, 1440, 1240, 12-760, 1400, 530, 1100,, 1160
1100	Dump Buffer	1400
1160	Release Job	1400
1240	Load Buffer	1400
1300	Enter CP Status	10-760, 12-760
1400	Call RPL Package	
1400	Request Tape	17-760, 12-760

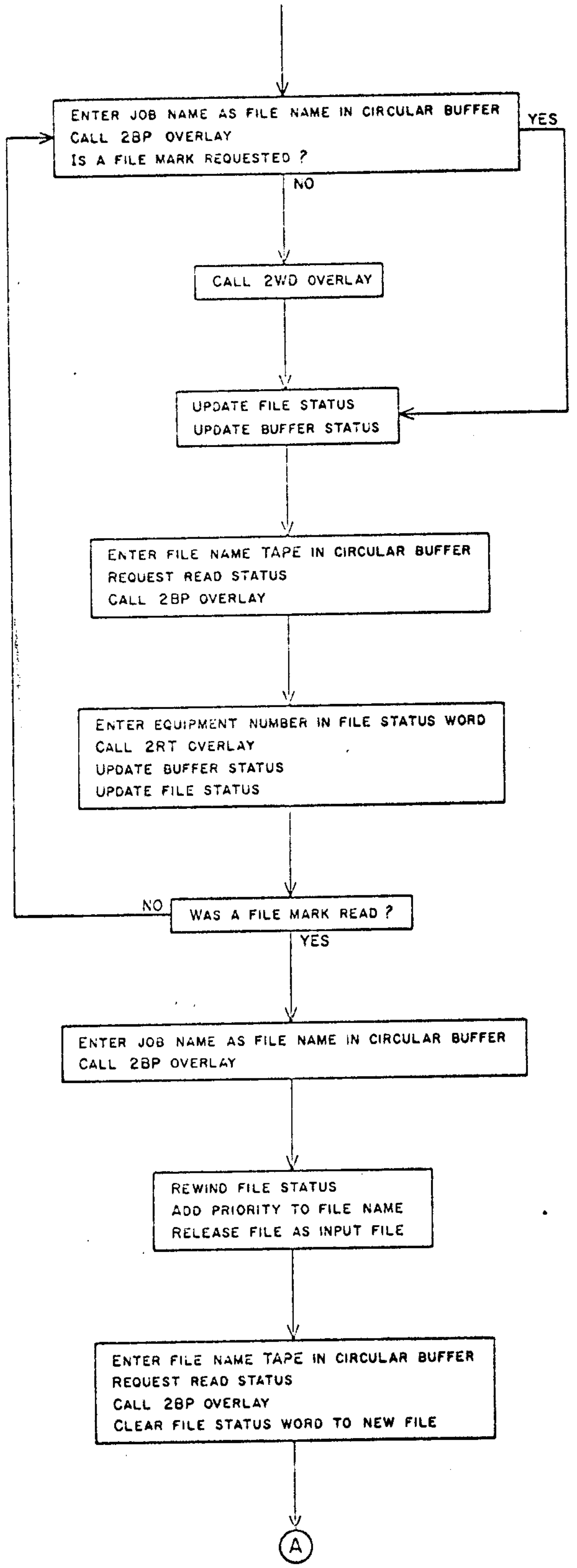
Direct Core Cells

P20/24	FST entry for file sent to 2BP
P30/34	Job name from job card set up by 2TJ
P35	Priority from job card set up by 2TJ
P40/44	File Control Word+Buffer Status (same as RA+3)
P50/54	Input Register
P55	RA from CP(20)
P56	FL from CP(20)
P57	FST entry address set up by 2BP
P70	0001 (constant)
P71	0100 (constant)
P72	1000 (constant)
P74	Control point address
P75	Input register address


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 10000B
READ RA AND FL
DOES FL=10000B ?
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 2BF 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)


ENTER JOB NAME AS FILE NAME IN CIRCULAR BUFFER
CALL 28P OVERLAY
IS A FILE MARK REQUESTED ?
YES
NO
CALL 2WD OVERLAY
UPDATE FILE STATUS
UPDATE BUFFER STATUS
ENTER FILE NAME TAPE IN CIRCULAR BUFFER
REQUEST READ STATUS
CALL 28P OVERLAY
ENTER EQUIPMENT NUMBER IN FILE STATUS WORD
CALL 2RT OVERLAY
UPDATE BUFFER STATUS
UPDATE FILE STATUS
NO
WAS A FILE MARK READ ?
YES
ENTER JOB NAME AS FILE NAME IN CIRCULAR BUFFER
CALL 28P OVERLAY
REWIND FILE STATUS
ADD PRIORITY TO FILE NAME
RELEASE FILE AS INPUT FILE
ENTER FILE NAME TAPE IN CIRCULAR BUFFER
REQUEST READ STATUS
CALL 28P OVERLAY
CLEAR FILE STATUS WORD TO NEW FILE
A

ROUTINE: 1TD - Phase 3 Tape Dump

PURPOSE: To dump completed output files on tape in order of priority for off-line printing.

GENERAL: 1TD is assigned a PP and a control point when "X.DUMP." is typed. Whenever all output files are dumped, the package is released.

METHOD:

1. "DUMP" is assigned as the job name for the control point. 10000_8 words of central memory are required for the buffer and if it is not assigned, the PP is released.

2. The message "REQUEST TAPE" appears as the third line of the control point. The operator must enter "X.ASSIGN YY.", where YY is a tape equipment number.

3. The FNT is searched for the highest priority output file.

4. The file is assigned to the control point as a local file and the job name from FNT is set into CP(21). The job name replaces "DUMP" at the control point and "DUMP" is displayed as the console message.

5. The central memory buffer is filled by 2RD.

6. 2WT is called and the tape equipment number is set in FST. If the tape assigned is ½", a return jump is made to the BCD write coding in 2WT. A 1" tape assignment giver the binary write of 2WT control.

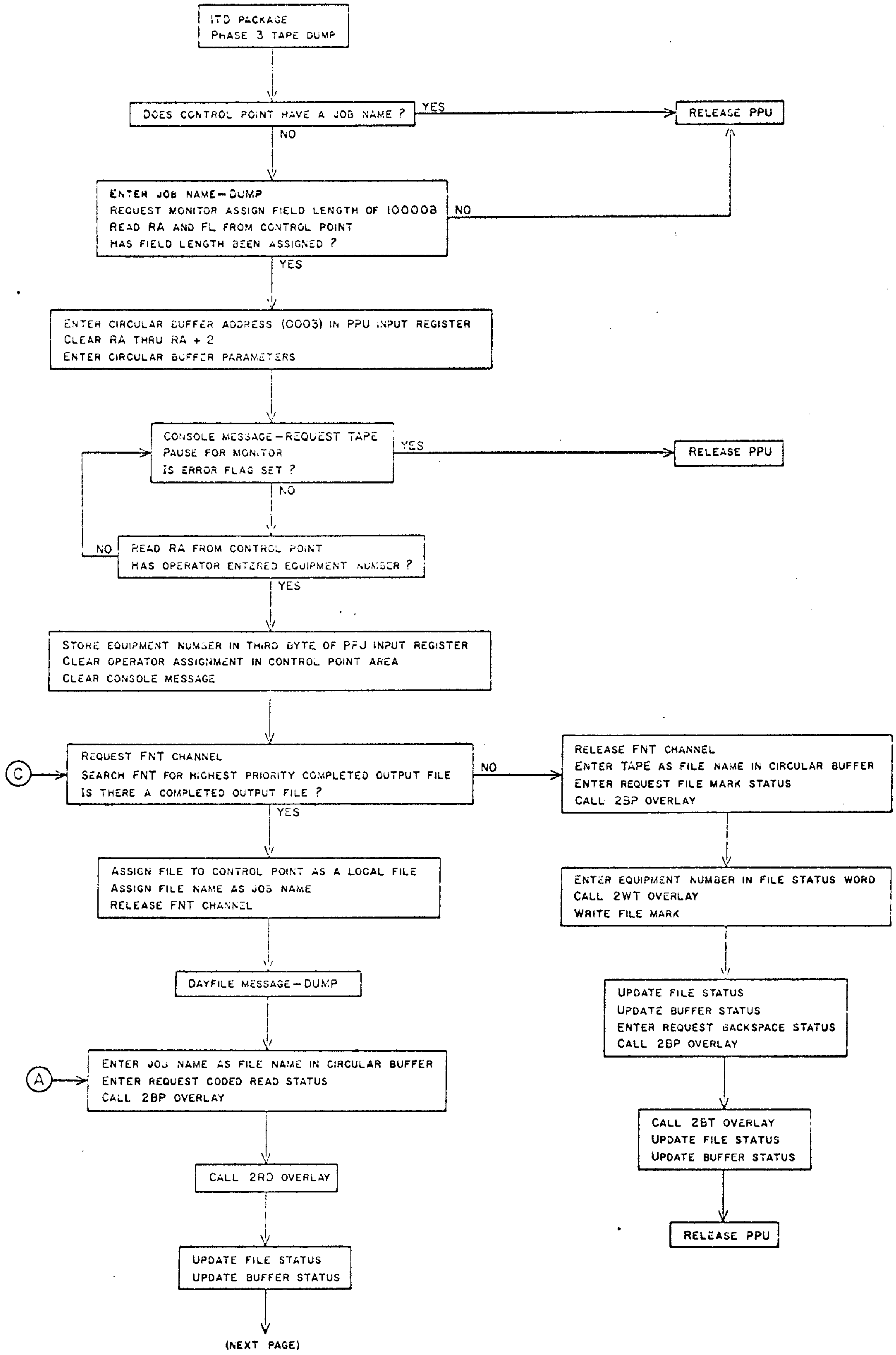
7. When the buffer is emptied the FST and buffer status are updated. No file mark is written between jobs.

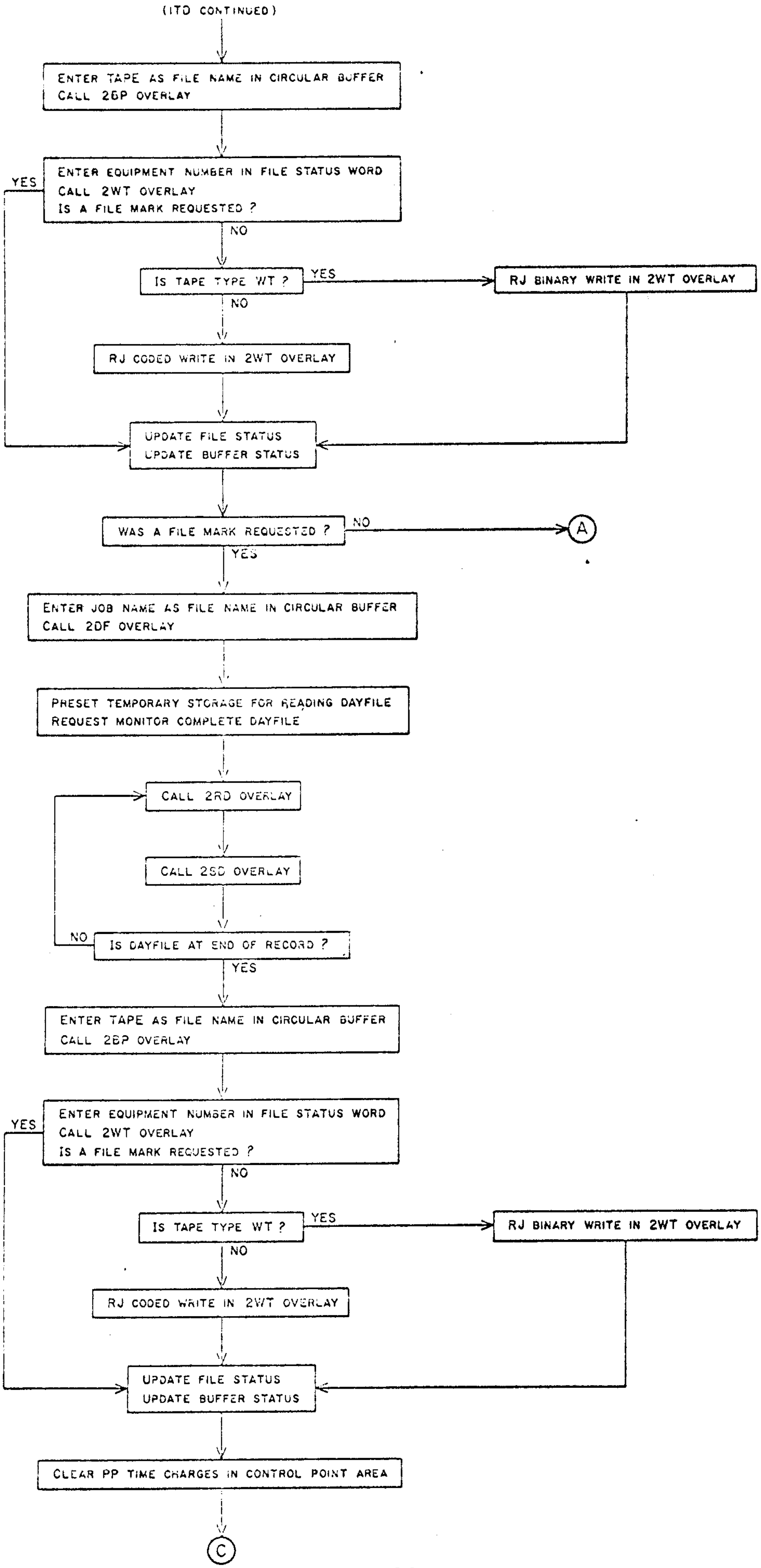
8. Whenever the job output file has been dumped and a file mark requested, 2DF is called to drop the disk tracks used by the file.

9. 2RD and 2SD search the dayfile for entries pertaining to the job and they are written after the job output by 2WT.

10. All PP time charges at the control point are cleared.

11. Again FNT is searched for the highest priority output file. When no more output files exist, a file mark is written and then the tape is backspaced over it. The tape is left in this position so that more dumps may be added.


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)


(ITO 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

PROGRAM: 2BP -- Read Buffer Parameter

PURPOSE: To examine the buffer arguments for correctness, enter file name in FNT, and reserve the file.

GENERAL: This routine is called by 1AJ, 1BJ, 1DJ, 1LJ, 1LT, 1TD, CIO to check the buffer arguments for range and validity. It also enters file name in the FNT, reserves the file if possible. The following error messages are produced: BUFFER ARG ERROR, and FNT LIMIT.

METHOD:

1. Read buffer status and arguments.

2. Move the arguments to a two word/entry table at P60.

3. Check for argument region out of field limit range. If in error, display in dayfile - BUFFER ARG ERROR, issue a FC of 13B (abort CP), and exit to PP monitor loop.

4. Check for LIMIT over field limit and go to the error procedure if it is.

5. Check for OUT $\geq$ LIMIT.

6. Check for IN $\geq$ LIMIT.

7. Check for OUT $<$ FIRST.

8. Check for IN $<$ FIRST.

9. Check each character of file name to first blank for less than 37. If an error is detected, go through same error procedure as above. Also senses inserted characters after the first blank as errors. Finally, it checks to make certain file name is non-blank.

10. Searches FNT for the file name and matching CP number. On a find, it saves FST entry address.

11. If the file was not found in FNT, it locks out other PP's from the FNT. A blank entry is found and the name is entered with its CP, file set as local, and priority of zero. A blank entry is written into FST. Channel 15 is released thereby allowing other PP's into FNT, and FST address is saved.

12. Request channel 14 (FST lock out channel). Check LBS field of FST for file reserved (even number - reserved). If it is not reserved, reserve it (set FST odd), release channel 14 and exit.

13. If it is reserved, release channel 14, and issue a 17B

function to allow the monitor to move central storage.

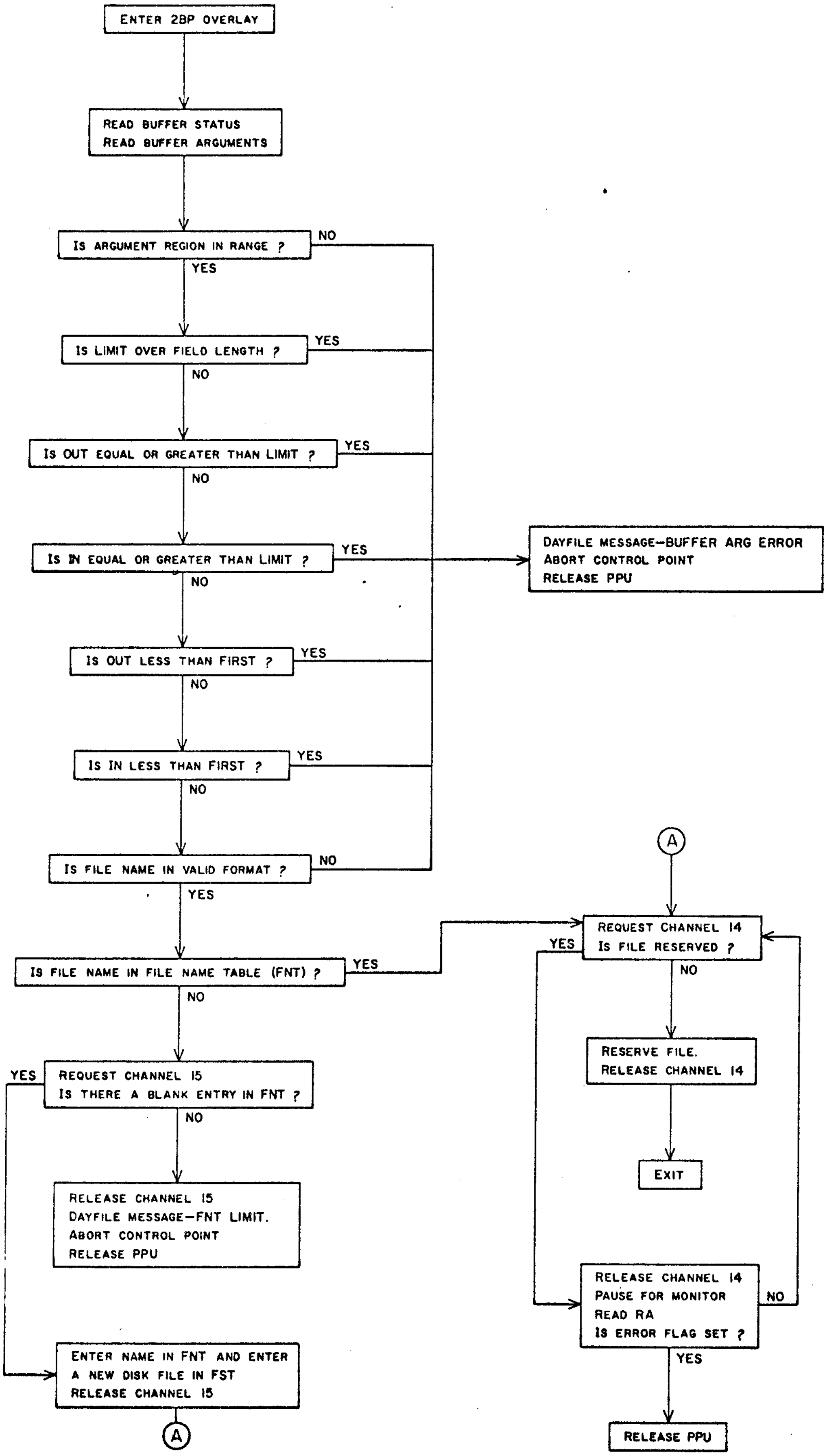
14. Read CP status. Save reference address. If error flag is not set, go back to No. 12 above and continue. If flag is set, release PP(12B) and exit to resident PP program.

2BP Routines

2000	Main Program	2350, 2300, 2150, 2100
2100	Alter File Status	14-740, 14-750, 17-760, 12-760, 100
2150	Search FNT	15-740, 15-750, 530, 13-760, 100
2300	Verify File Name	530, 13-760, 100
2350	Verify Argument Values	530, 13-760, 100

Direct Core Cells

2000	P50/54	Input register (Buffer address)
	P40/44	Status
	P01	Counter for buffer parameters
	P02	Address for storing buffer arguments
	P60/70	Buffer arguments (FIRST, IN, OUT, LIMIT)
	P10/14	Temporary storage for buffer arguments
2100	P57	File status address
	P20/24	File status
	P45	Last buffer status from FST
	P40/44	Buffer status
	P74	Control Point address
	P10/14	CP status
	P55	Reference address
2150	P20/24	FNT address and limit
	P10/14	FNT entry
	P40/44	Buffer status (Name of file)
	P51	Input register (CP for file)
	P57	File status address
2300	P01	Address of file name
	P40/44	Buffer status (file name)
2350	P53/54	Argument address
	P56	Field length
	P60/61	FIRST
	P62/63	IN
	P64/65	OUT
	P66/67	LIMIT


ENTER 2BP OVERLAY
READ BUFFER STATUS
READ BUFFER ARGUMENTS
IS ARGUMENT REGION IN RANGE ?
NO
YES
IS LIMIT OVER FIELD LENGTH ?
YES
NO
IS OUT EQUAL OR GREATER THAN LIMIT ?
YES
NO
IS IN EQUAL OR GREATER THAN LIMIT ?
YES
NO
DAYFILE MESSAGE—BUFFER ARG ERROR
ABORT CONTROL POINT
RELEASE PPU
IS OUT LESS THAN FIRST ?
YES
NO
IS IN LESS THAN FIRST ?
YES
NO
IS FILE NAME IN VALID FORMAT ?
NO
YES
IS FILE NAME IN FILE NAME TABLE (FNT) ?
YES
NO
REQUEST CHANNEL 15
IS THERE A BLANK ENTRY IN FNT ?
YES
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

ROUTINE: 2BT - BACKSPACE TAPE

PURPOSE: To backspace a block of binary or BCD data on tape and set the buffer addresses accordingly.

GENERAL: 2BT is called by CIO when a backspace request on a tape unit is received. All backspacing over logical records in the buffer is assumed to be completed by the calling program.

METHOD: A. BINARY

1. If a binary backspace is requested, two blocks are backspaced and then the last one backspaced is read.

2. This block is checked for a short block; if it is short, IN and OUT are set equal to FIRST.

3. If it is not short, the backing of two blocks and reading of one is continued until a short block is found.

B. BCD

1. If the tape type is MT ($\frac{1}{2}$"), one block is backed and IN and OUT are set equal to FIRST. One BCD record (normally a card image or print line) will be backspaced in this case.

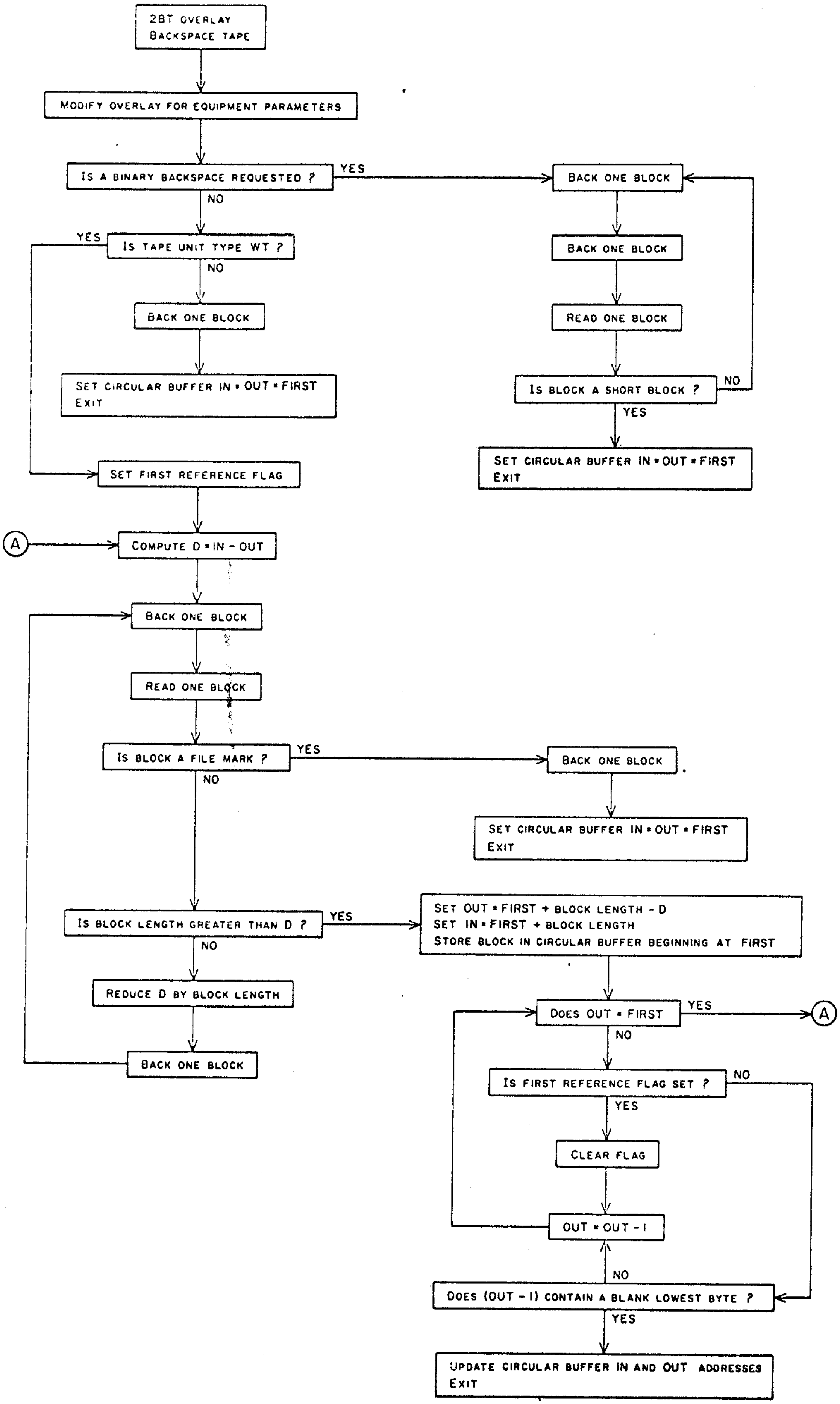
2. A BCD backspace on a 1" tape begins with the computation of the amount of data left in the buffer as a result of the last read. This quantity, referred to as D, is equal to IN-OUT. This data was left in the buffer as a result of the last read, and may have been stored on the disk in several sectors. The system assumes that the calling program will backspace within the buffer, and so, before beginning a logical BCD record backspace on the tape, 2BT will backspace the tape a number of blocks equivalent to the amount of data contained in the buffer. This quantity is represented by D.

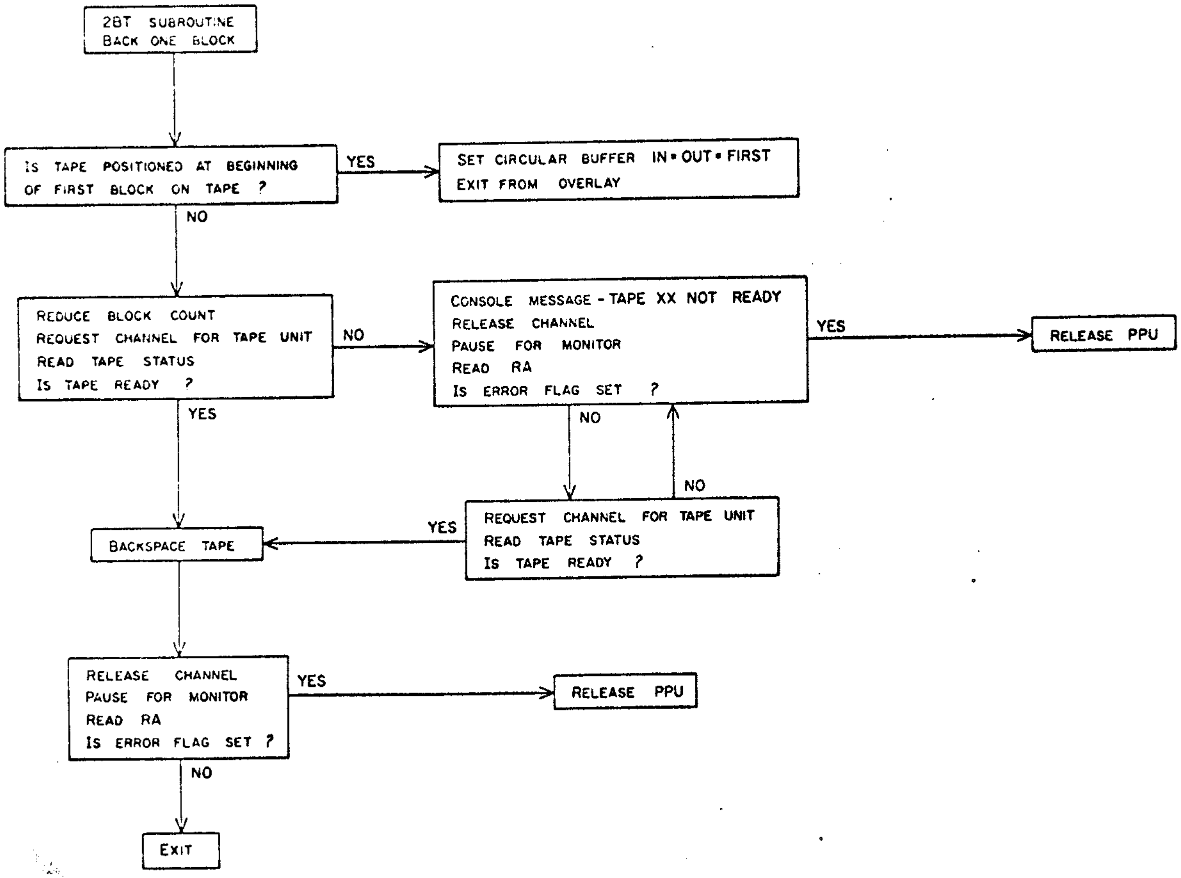
3. 2BT therefore backspaces over a block and reads that block into peripheral processor memory. The block length is then compared with D: if less than D, then this block is assumed to contain data which already has been read into the buffer. 2BT then decreases D by this amount, backspaces over this block and the block preceding it, and then reads the block. The process of backspacing, reading, and reducing D is repeated until a block is read whose length is greater than the present value of D;

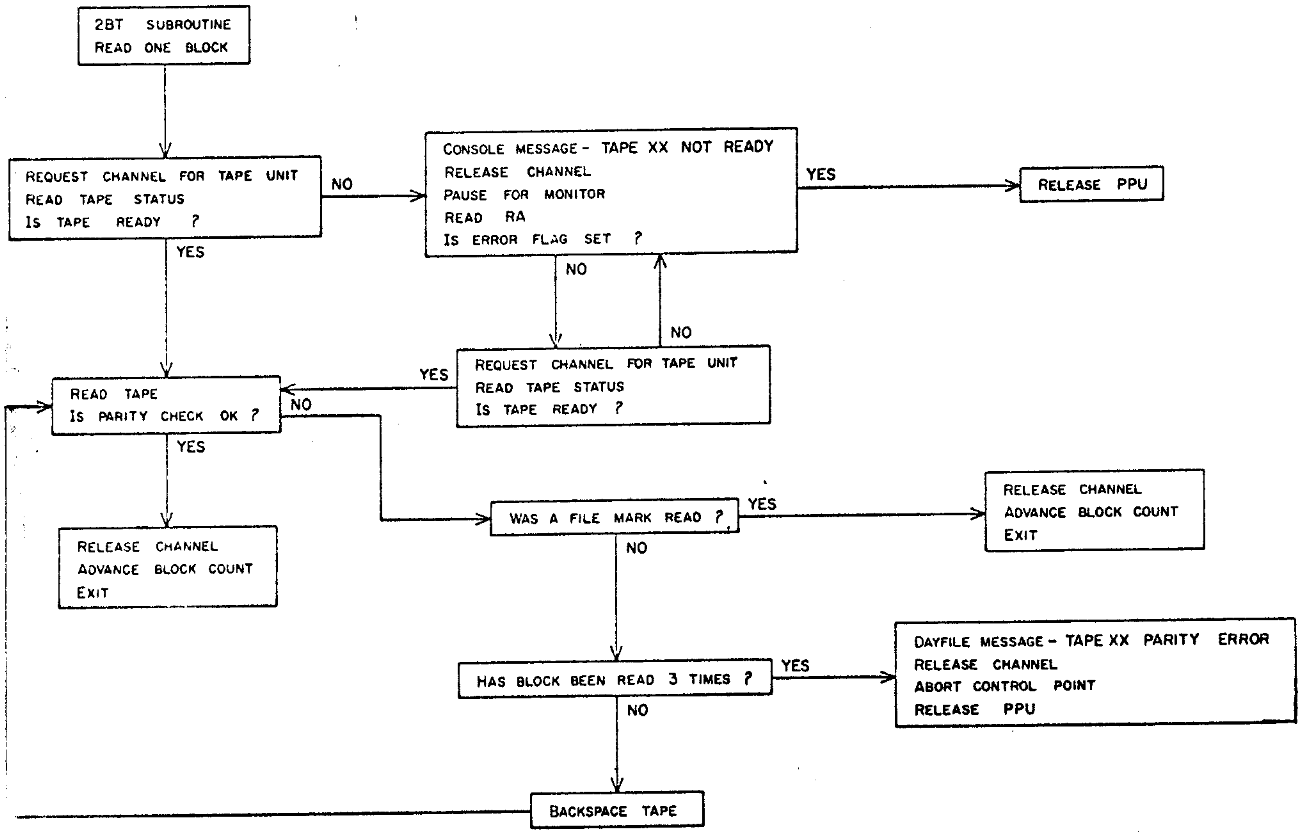
4. 2BT transfers this block from peripheral processor memory to the circular buffer beginning at FIRST. If D is still non-zero, then part of this block contains data residing in the buffer at the time the backspace was requested, and presumably has been searched by the calling program.: 2BT there fore sets the OUT pointer to FIRST + block length - D. At the same time, the IN pointer is set to reflect the transfer of the sector to the buffer.

5. 2BT then searches each word in the buffer from OUT - 1 down to FIRST until a word with a zero low-order byte is found, indicating the end of a logical BCD record. When the end of the record is found, 2BT updates the IN and OUT pointers in the calling'programs' argument list, and returns control to CIO.

6. OUT now points to the first word following the end of the logical record. If no zero low-order byte was found then 2BT backspaces two blocks and reads one, and then repeats the buffer search.


2BT OVERLAY
BACKSPACE TAPE
MODIFY OVERLAY FOR EQUIPMENT PARAMETERS
IS A BINARY BACKSPACE REQUESTED ?
YES
NO
IS TAPE UNIT TYPE WT ?
YES
NO
BACK ONE BLOCK
SET CIRCULAR BUFFER IN = OUT = FIRST
EXIT
SET FIRST REFERENCE FLAG
BACK ONE BLOCK
BACK ONE BLOCK
READ ONE BLOCK
IS BLOCK A SHORT BLOCK ?
NO
YES
SET CIRCULAR BUFFER IN = OUT = FIRST
EXIT
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
NO
NO
YES
REQUEST CHANNEL FOR TAPE UNIT
READ TAPE STATUS
IS TAPE READY ?
YES
BACKSPACE TAPE
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
NO
NO
YES
REQUEST CHANNEL FOR TAPE UNIT
READ TAPE STATUS
IS TAPE READY ?
YES
READ TAPE
IS PARITY CHECK OK ?
NO
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

ROUTINE: 2EF -- Process Error Flag

PURPOSE: To determine type of error and set up to execute the group of control cards after the EXIT. statement if one exits.

GENERAL: 2EF is called by the Advance Job routine (1AJ) when the error flag is sensed set (non-zero).

METHOD:

1. Read the control point status word from CP(20). Clears the error flag and stores status back to CP(20).

2. Uses the error flag to pick up address of error message. (Error Flag 1- Time Limit, 2- Arithmetic Error, 5- PP Call Error, 6- Operator Drop, 7- Track Limit)

3. Dayfile message routine is called to enter error message if the error condition was one of the above.

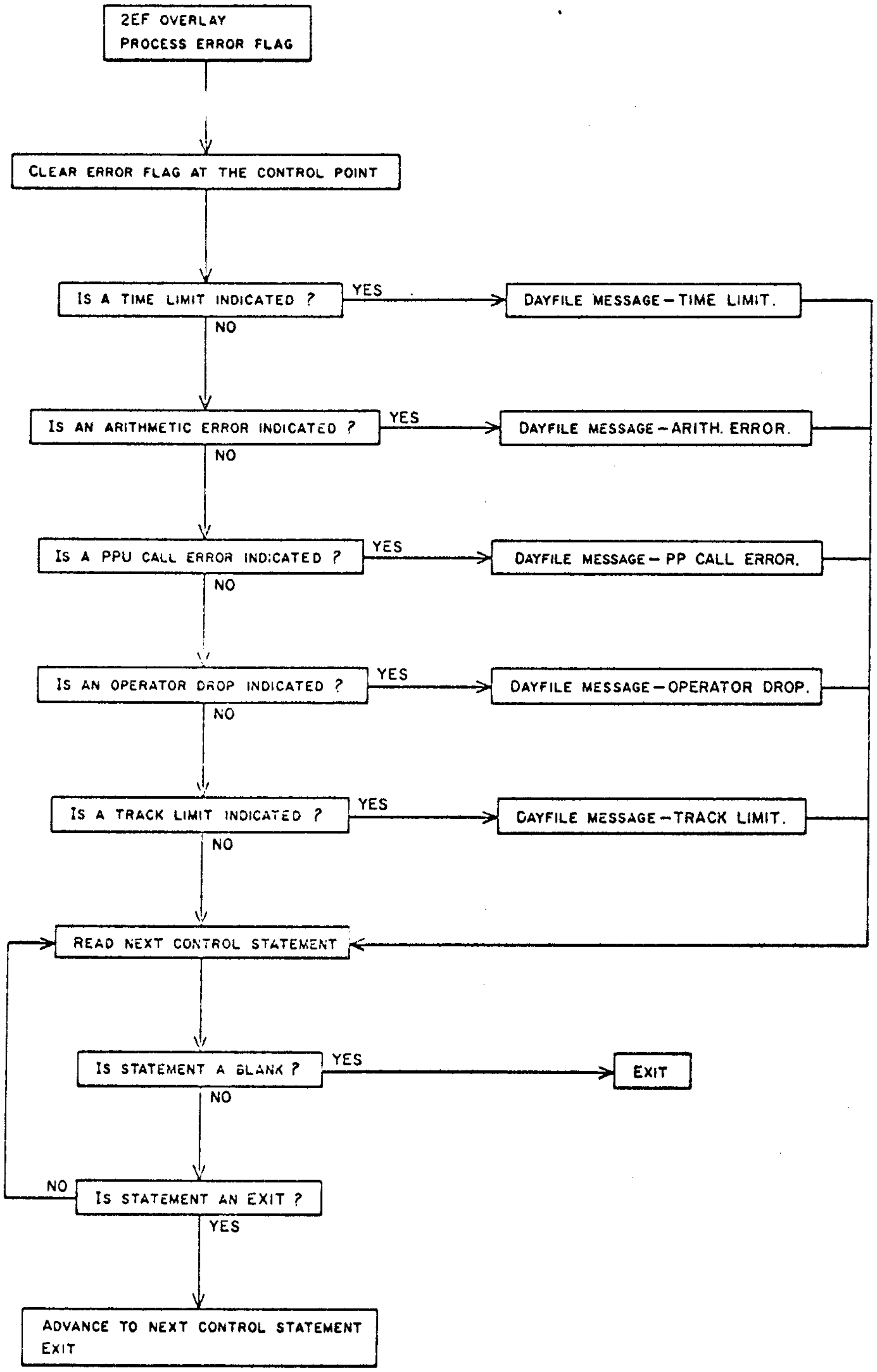
4. Control statements are searched untilthe last one is read or an EXIT. statement is encountered. The statement address at CP(21) is set to point to either the end of the statement list or the statement after the EXIT. card.

2EF Routines

2000	Main Program	2030, 2100, 531
2030	Error Table	
2100	Search for Exit	

Direct Core Cells

2000	P74	CP address
	P10/14	CP status
	P01	Error Flag
2100	P74	CP address
	P20/24	Next statement address
	P10/14	CP status


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

ROUTINE: 2LP -- PRINT

PURPOSE: To transfer data from the circular buffer to the line printer.

GENERAL: 2LP is called by the CIO Write Function routine once the file type has been determined to be a line printer.

METHOD:

1. A check is made for data left in the buffer. If there is none and the end-of-record was requested, IN and OUT are set equal to first, and EXIT is taken.

2. If there is data, a word is read up and copied into the print line buffer. If the lowest byte of the word is not zero and 120 characters have not been assembled, another word is fetched.

3. If either a zero byte is found or 120 characters have been assembled, a transfer is made to location 2150 (PRINT LINE). This subroutine finds an available printer and prints the line.

4. Three characters are checked in the first character position for carriage control:

   (0) - advance paper one extra line after printing.

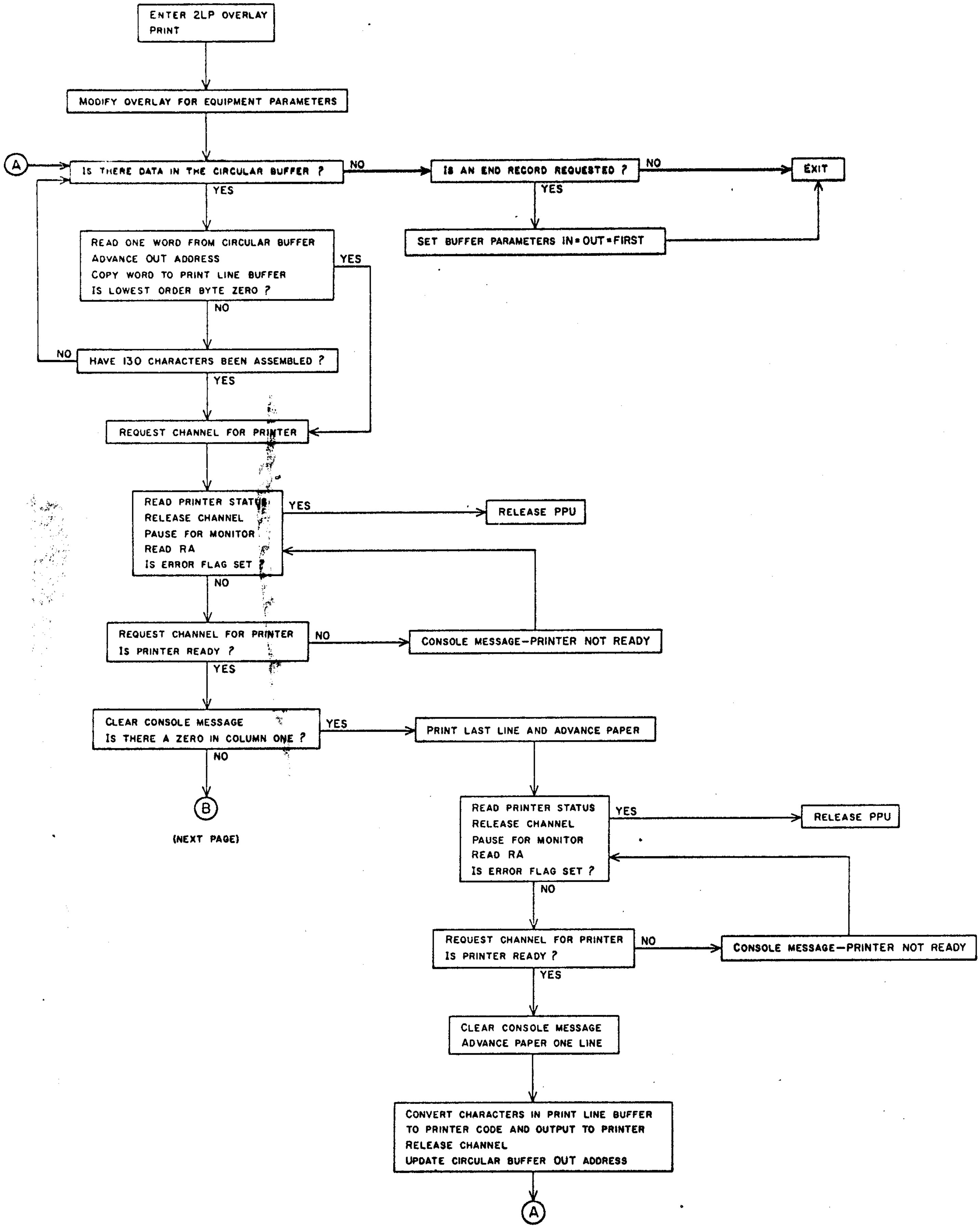
   (1) - advance to top of form after printing line.

   (+) - print the last line but do not advance the paper.

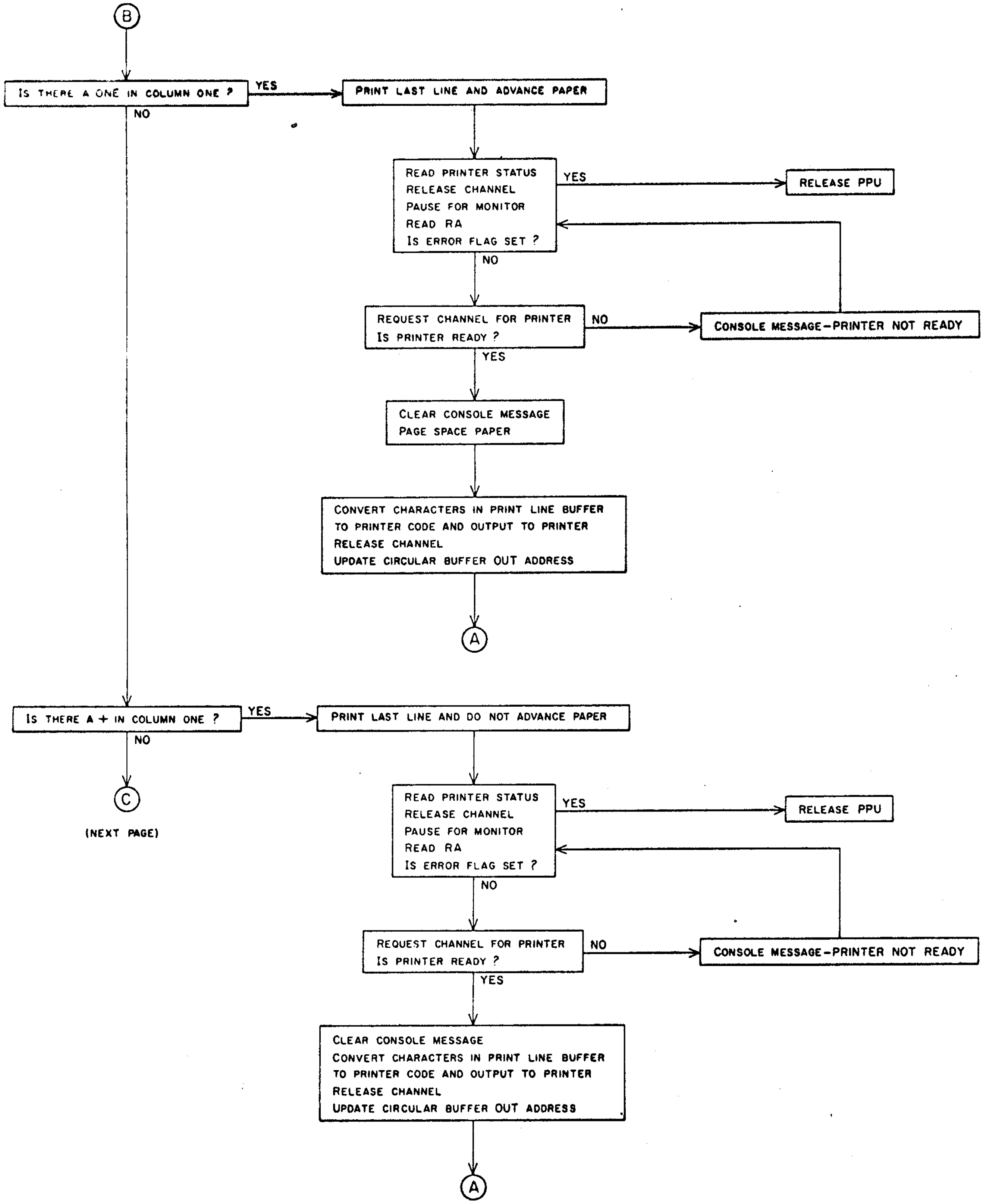
5. If there is a 7X code in column one, the last line is printed, and then printer carriage control X is selected.

6. If none of the above mentioned codes are in column one, the line is printed and the paper advanced.

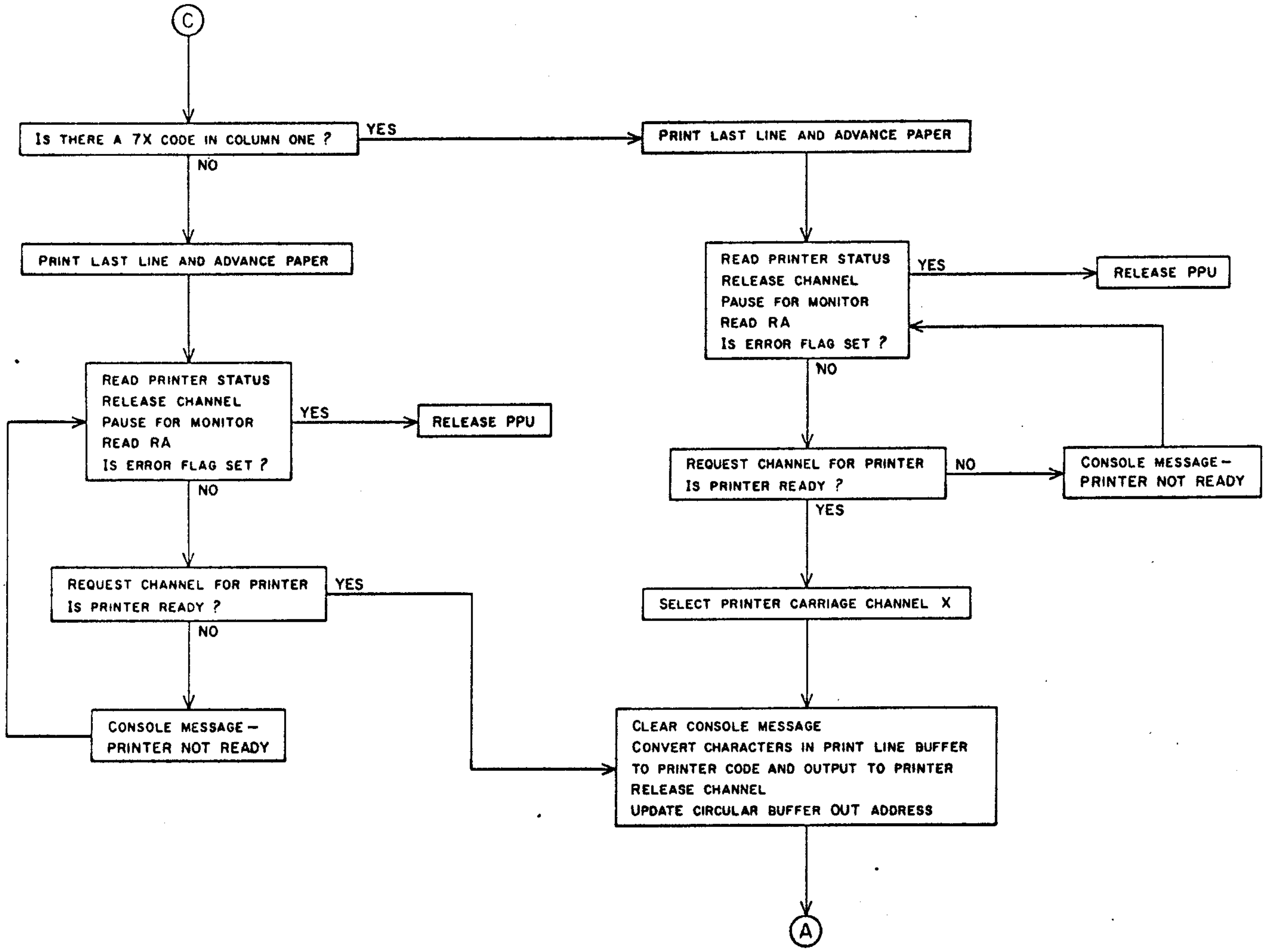
7. 2LP then returns to its beginning routine to check if any data is left in the buffer.


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
SET BUFFER PARAMETERS IN=OUT=FIRST
YES
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)


B
IS THERE A ONE IN COLUMN ONE ?
YES
NO
PRINT LAST LINE AND ADVANCE PAPER
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
PAGE SPACE PAPER
CONVERT CHARACTERS IN PRINT LINE BUFFER
TO PRINTER CODE AND OUTPUT TO PRINTER
RELEASE CHANNEL
UPDATE CIRCULAR BUFFER OUT ADDRESS
A
IS THERE A + IN COLUMN ONE ?
YES
NO
PRINT LAST LINE AND DO NOT ADVANCE PAPER
C
(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
CONVERT CHARACTERS IN PRINT LINE BUFFER
TO PRINTER CODE AND OUTPUT TO PRINTER
RELEASE CHANNEL
UPDATE CIRCULAR BUFFER OUT ADDRESS
A

(2LP CONTINUED)


C
IS THERE A 7X CODE IN COLUMN ONE ?
YES
NO
PRINT LAST LINE AND ADVANCE PAPER
PRINT LAST LINE AND ADVANCE PAPER
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 ?
YES
NO
CONSOLE MESSAGE – PRINTER NOT READY
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
SELECT PRINTER CARRIAGE CHANNEL X
CLEAR CONSOLE MESSAGE
CONVERT CHARACTERS IN PRINT LINE BUFFER
TO PRINTER CODE AND OUTPUT TO PRINTER
RELEASE CHANNEL
UPDATE CIRCULAR BUFFER OUT ADDRESS
A

ROUTINE: 2PC -- Punch Cards

PURPOSE: To punch either binary or Hollerith cards.

GENERAL: 2PC is called by the CIO Write Function routine once the file type has been determined to be a card punch.

METHOD:

1. A check is made for a request to punch Hollerith. A return jump is made to either PUNCH BINARY or PUNCH BCD.

   A. PUNCH BINARY

      1. If there is enough data for a full card, the punch buffer is cleared for 15 words.

         a) The data is transferred to the punch buffer.

         b) The card length is set in column one.

         c) The checksum set in column two.

         d) The card count is advanced.

         e) The card count is entered in column 80.

      2. A channel is then requested, with a "PUNCH NOT READY" message displayed if needed. If the punch is ready, a card is punched, the channel released, OUT is updated, and a check for the error flag in RA is made.

      3. If an error exists, the PPU is released.

      4. If there is no error, a check is again made to see if there is enough data for a full card. If there is not enough data for a full card and an end-of-record is selected, the partial card will be punched.

      5. If there is no data, a 7-8-9 card is punched, and IN and OUT are set equal to FIRST.

   B. PUNCH BCD

      1. The punch buffer is cleared for 80 characters.

      2. If there is at least one word left in the buffer, the word is converted into 10 Hollerith characters.

      3. A check is always made to see if 80 characters have been assembled. If not, a check for a lowest order byte of zero is made. If it is not present, another word is assembled.

4. If either 80 characters have been assembled or a zero byte is found, the card is punched.

5. The error flag is then checked in RA and the PP is released if an error exists. If there is no error, OUT is updated, the card count advanced, and a return is made to convert another 80 characters.

6. If there is not another full word in the buffer and a file mark is requested:

   a) The card count is cleared.

   b) A 6-7-8-9 card is punched.

   c) IN and OUT set equal to FIRST, and

   d) An EXIT taken.

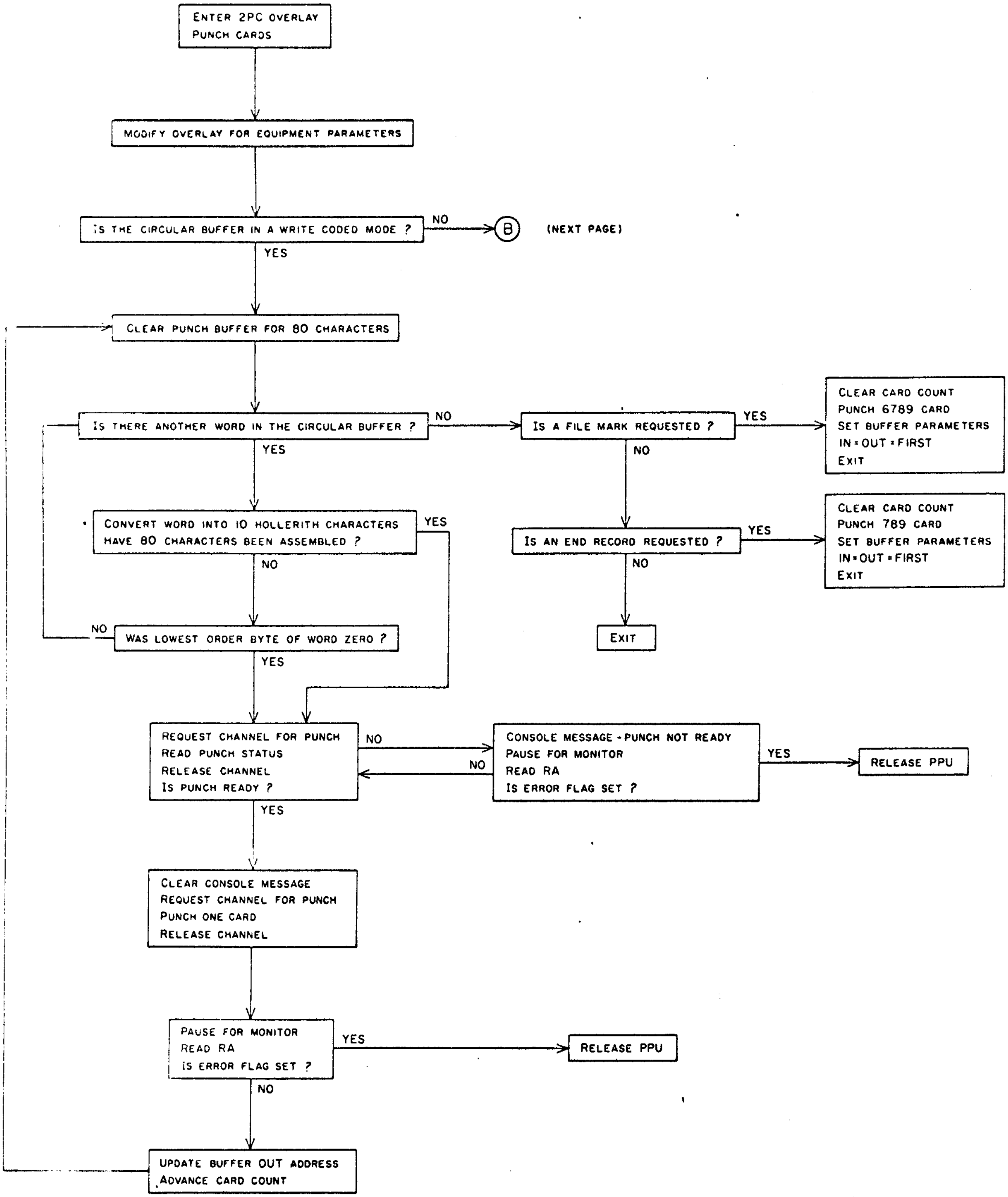
7. If an end-of-record is requested,

   a) A 7-8-9 card is punched.

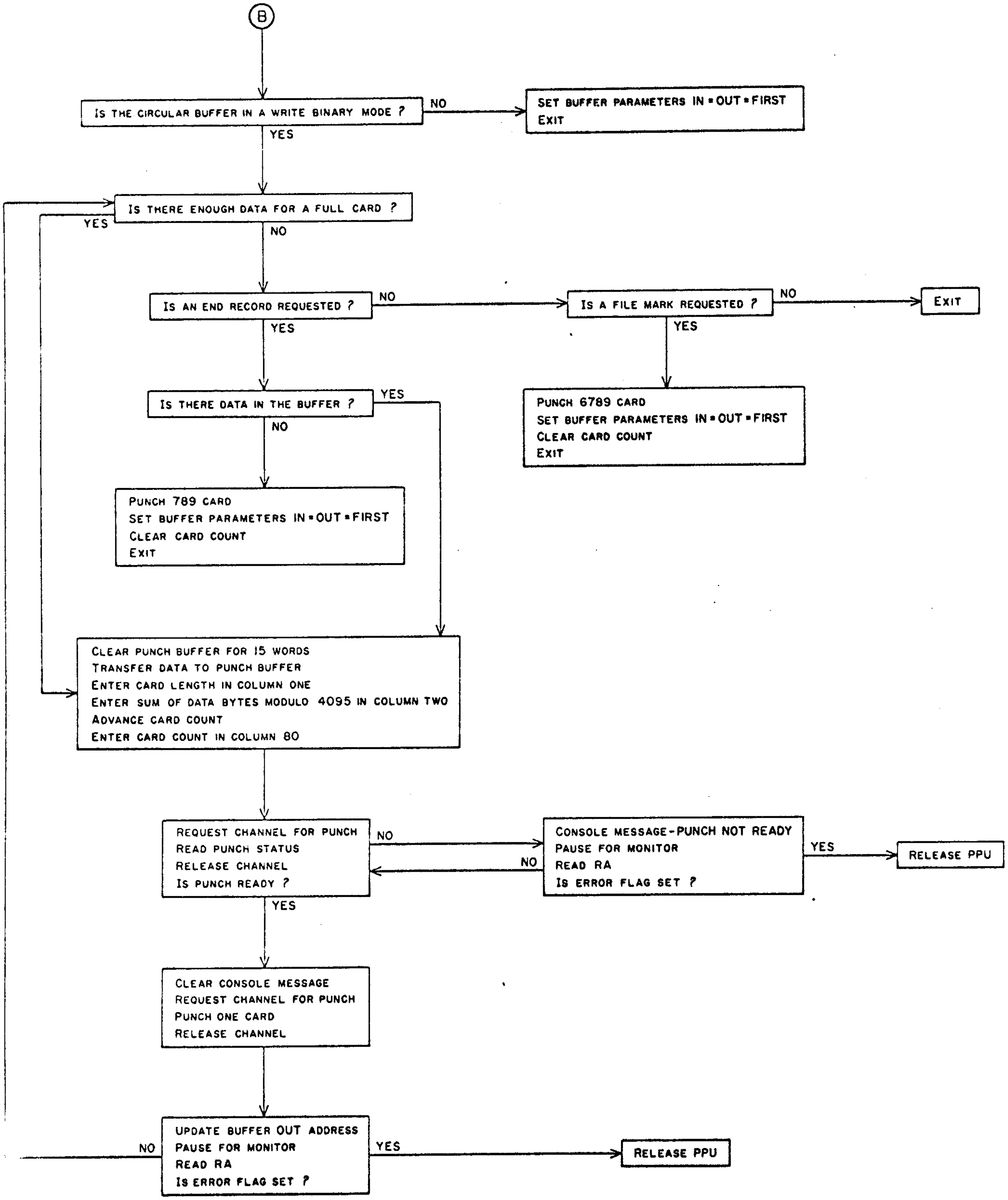
   b) The card count is cleared.

   c) IN and OUT set equal to FIRST, and

   d) EXIT taken.


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
YES
NO
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
NO
WAS LOWEST ORDER BYTE OF WORD ZERO ?
EXIT
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 ?
NO
YES
RELEASE PPU
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)


B
IS THE CIRCULAR BUFFER IN A WRITE BINARY MODE ?
NO
SET BUFFER PARAMETERS IN = OUT = FIRST
EXIT
YES
IS THERE ENOUGH DATA FOR A FULL CARD ?
YES
NO
IS AN END RECORD REQUESTED ?
NO
IS A FILE MARK REQUESTED ?
NO
EXIT
YES
PUNCH 6789 CARD
SET BUFFER PARAMETERS IN = OUT = FIRST
CLEAR CARD COUNT
EXIT
YES
IS THERE DATA IN THE BUFFER ?
YES
NO
PUNCH 789 CARD
SET BUFFER PARAMETERS IN = OUT = FIRST
CLEAR CARD COUNT
EXIT
CLEAR PUNCH BUFFER FOR 15 WORDS
TRANSFER DATA TO PUNCH BUFFER
ENTER CARD LENGTH IN COLUMN ONE
ENTER SUM OF DATA BYTES MODULO 4095 IN COLUMN TWO
ADVANCE CARD COUNT
ENTER CARD COUNT IN COLUMN 80
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 ?
NO
YES
RELEASE PPU
YES
CLEAR CONSOLE MESSAGE
REQUEST CHANNEL FOR PUNCH
PUNCH ONE CARD
RELEASE CHANNEL
UPDATE BUFFER OUT ADDRESS
PAUSE FOR MONITOR
READ RA
IS ERROR FLAG SET ?
NO
YES
RELEASE PPU

ROUTINE: 2RC - Read Cards

PURPOSE: To read cards from the card reader and process them either as binary or BCD cards.

GENERAL: 2RC is called by the CIO Read Function routine once the file type has been determined to be a card reader.

METHOD:

1. If the End-of-Job flag is set, 2RC clears the flag, sets the file mark and exits.

2. A check is made to see if the buffer has room for 15 words of input. If not, an EXIT is taken and no read is performed.

3. A return jump is taken to READ NEXT CARD which requests the correct channel, makes sure the reader is ready, and reads the next card.

4. Once a card is read, the card count is advanced in the FST entry and a check is made for 7-8-9 punches in column one. If there are only 7-9 punches, a transfer is made to PROCESS BINARY CARD. If neither condition exists, PROCESS HOLLERITH CARD is given control.

5. After the card is processed, the IN address of the central memory buffer is incremented by the number of words read.

6. Another card is then read if the buffer length allows it and there are no errors.

7. If a 7-8-9 card was found, an end-of-record indicator is set and the card count is cleared. An EXIT is then made.

8. If a 6-7-8-9 card was found:
   a) and the last record was not complete, the End-of-Job flag is set along with End-of-record. The next time through 2RC, the EOJ flag will be cleared and a file mark will be written.

   b) and the last record was complete, the file mark indicator is set if the buffer is empty.

   c) and the last record was complete, the EOJ flag and End-of-record indicator are set if the buffer is not empty.

A. PROCESS BINARY CARD

1. The number of significant columns is determined from the word count in column one.

2. If there is a correction punch in column one, the significant words are copied into the circular buffer, IN is advanced and an EXIT taken.

3. Otherwise, the checksum is cleared and the column index is set to 2. Each significant column is then added to the checksum module 4095. If the checksum is zero, the significant words are copied into the buffer and IN is advanced.

4. If the checksum is not zero, a binary card error is displayed. After a 4 second delay, a check is made to see if the card reader is ready. If it is not, then the operator is given a chance to reread the card.

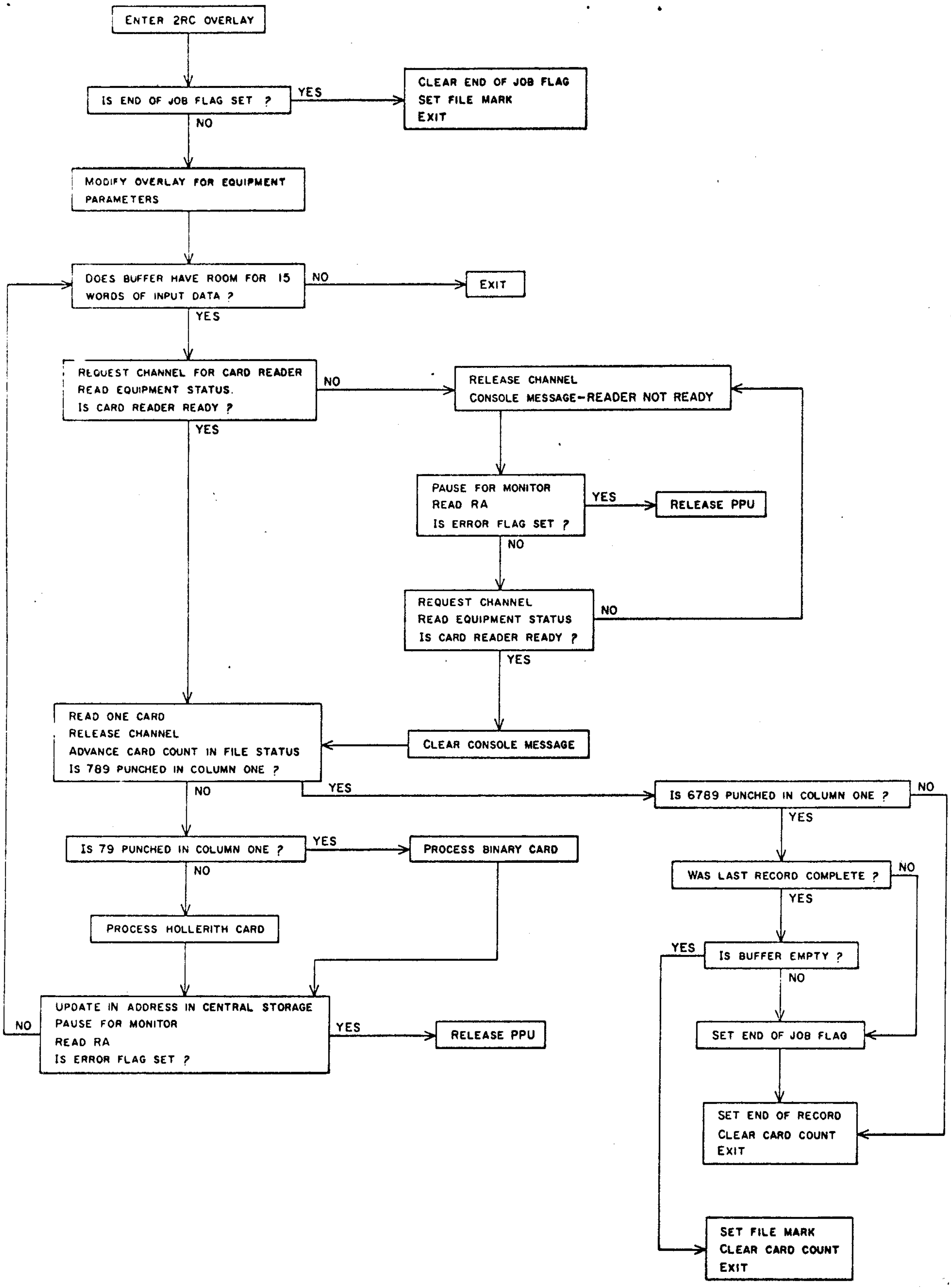
5. If the reader is ready, a check of the error flag in RA is made. If the error flag is not set, then the binary card error is displayed again.

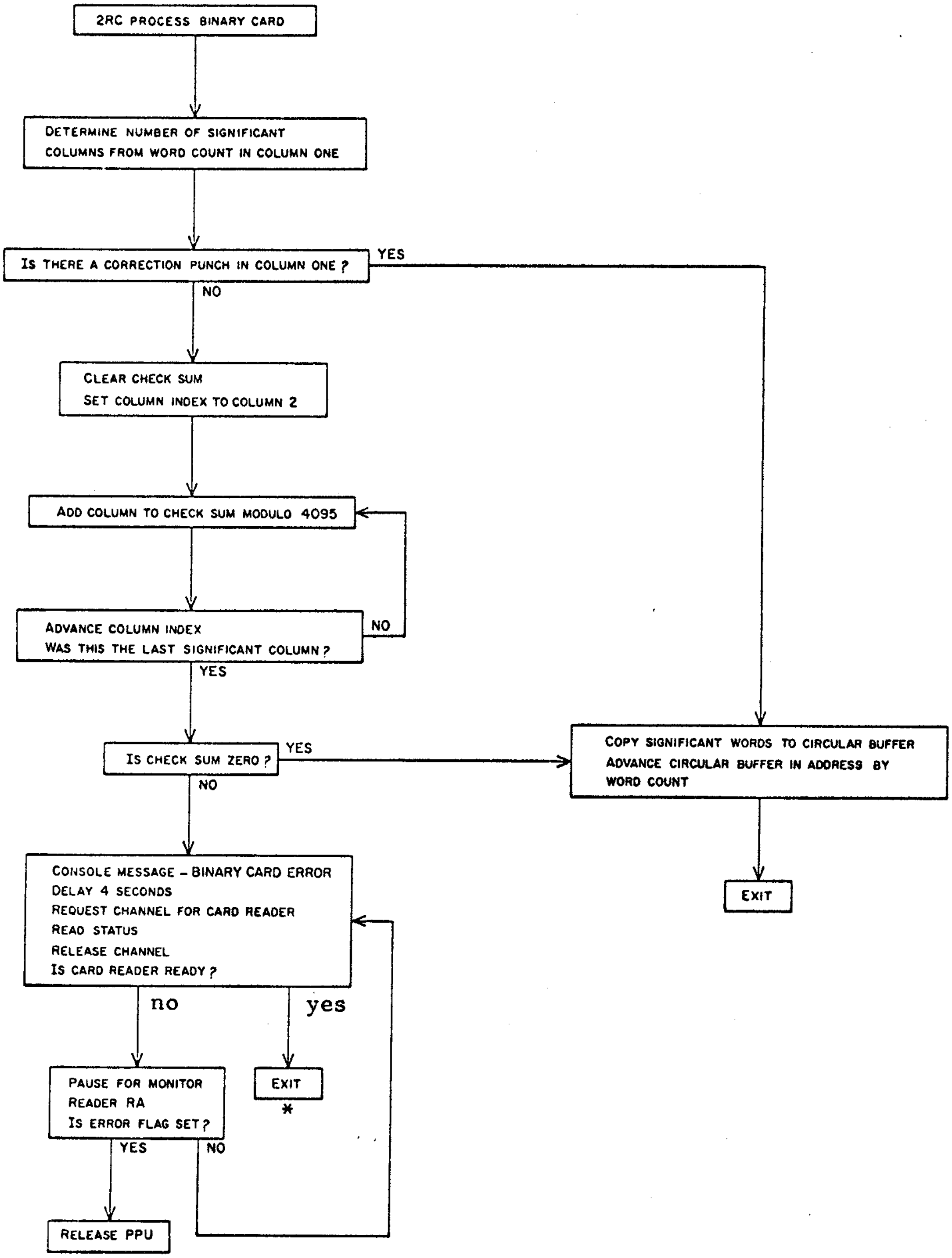
B. PROCESS HOLLERITH CARD

1. The last significant column is determined.

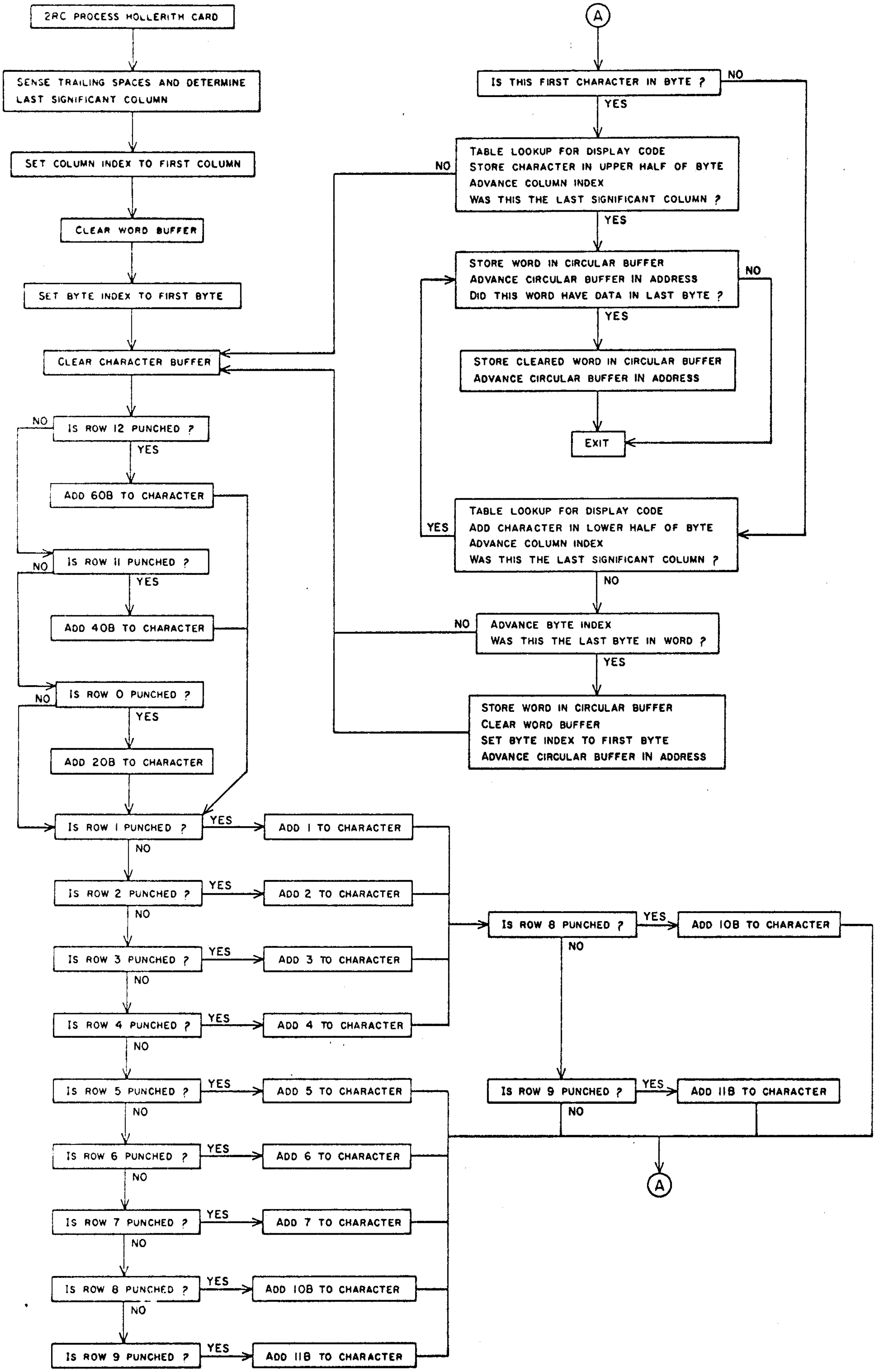
2. A table look-up is then done on each character to change the Hollerith character into display code. The significant characters are stored in the buffer by advancing IN.

3. If the last word's last byte has significant data, a cleared word is stored after it. If not, the last byte will be cleared.


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
CLEAR CONSOLE MESSAGE
READ ONE CARD
RELEASE CHANNEL
ADVANCE CARD COUNT IN FILE STATUS
IS 789 PUNCHED IN COLUMN ONE ?
YES
IS 6789 PUNCHED IN COLUMN ONE ?
NO
YES
NO
IS 79 PUNCHED IN COLUMN ONE ?
YES
PROCESS BINARY CARD
NO
PROCESS HOLLERITH CARD
WAS LAST RECORD COMPLETE ?
NO
YES
YES
IS BUFFER EMPTY ?
NO
NO
UPDATE IN ADDRESS IN CENTRAL STORAGE
PAUSE FOR MONITOR
READ RA
IS ERROR FLAG SET ?
YES
RELEASE PPU
SET END OF JOB FLAG
SET END OF RECORD
CLEAR CARD COUNT
EXIT
SET FILE MARK
CLEAR CARD COUNT
EXIT


2RC PROCESS BINARY CARD
DETERMINE NUMBER OF SIGNIFICANT COLUMNS FROM WORD COUNT IN COLUMN ONE
IS THERE A CORRECTION PUNCH IN COLUMN ONE ?
YES
NO
CLEAR CHECK SUM
SET COLUMN INDEX TO COLUMN 2
ADD COLUMN TO CHECK SUM MODULO 4095
ADVANCE COLUMN INDEX
WAS THIS THE LAST SIGNIFICANT COLUMN ?
NO
YES
IS CHECK SUM ZERO ?
YES
NO
COPY SIGNIFICANT WORDS TO CIRCULAR BUFFER
ADVANCE CIRCULAR BUFFER IN ADDRESS BY WORD COUNT
EXIT
CONSOLE MESSAGE – BINARY CARD ERROR
DELAY 4 SECONDS
REQUEST CHANNEL FOR CARD READER
READ STATUS
RELEASE CHANNEL
IS CARD READER READY ?
no
yes
PAUSE FOR MONITOR
READER RA
IS ERROR FLAG SET ?
EXIT
*
YES
NO
RELEASE PPU

* 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 ?
NO
YES
ADD 60B TO CHARACTER
IS ROW 11 PUNCHED ?
NO
YES
ADD 40B TO CHARACTER
IS ROW 0 PUNCHED ?
NO
YES
ADD 20B TO CHARACTER
IS ROW 1 PUNCHED ?
YES
ADD 1 TO CHARACTER
NO
IS ROW 2 PUNCHED ?
YES
ADD 2 TO CHARACTER
NO
IS ROW 3 PUNCHED ?
YES
ADD 3 TO CHARACTER
NO
IS ROW 4 PUNCHED ?
YES
ADD 4 TO CHARACTER
NO
IS ROW 5 PUNCHED ?
YES
ADD 5 TO CHARACTER
NO
IS ROW 6 PUNCHED ?
YES
ADD 6 TO CHARACTER
NO
IS ROW 7 PUNCHED ?
YES
ADD 7 TO CHARACTER
NO
IS ROW 8 PUNCHED ?
YES
ADD 10B TO CHARACTER
NO
IS ROW 9 PUNCHED ?
YES
ADD 11B TO CHARACTER
IS ROW 8 PUNCHED ?
YES
ADD 10B TO CHARACTER
NO
IS ROW 9 PUNCHED ?
YES
ADD 11B TO CHARACTER
NO
A
A
IS THIS FIRST CHARACTER IN BYTE ?
NO
YES
TABLE LOOKUP FOR DISPLAY CODE
STORE CHARACTER IN UPPER HALF OF BYTE
ADVANCE COLUMN INDEX
WAS THIS THE LAST SIGNIFICANT COLUMN ?
NO
YES
STORE WORD IN CIRCULAR BUFFER
ADVANCE CIRCULAR BUFFER IN ADDRESS
DID THIS WORD HAVE DATA IN LAST BYTE ?
NO
YES
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 ?
YES
NO
ADVANCE BYTE INDEX
WAS THIS THE LAST BYTE IN WORD ?
NO
YES
STORE WORD IN CIRCULAR BUFFER
CLEAR WORD BUFFER
SET BYTE INDEX TO FIRST BYTE
ADVANCE CIRCULAR BUFFER IN ADDRESS

ROUTINE: 2RT -- Read Tape

PURPOSE: To read binary and BCD data from magnetic tape or rewind the tape.

GENERAL: This package is called by the CIO Read Function when a magnetic tape is to be read. Control is transferred by CIO to one of three locations within 2RT:

a) READ BINARY TAPE

b) READ BCD TAPE

c) REWIND

METHOD: A. READ BINARY TAPE

1. There must be room in the buffer for a full block (1000_8 words) of data or no reading is done.

2. The requested tape unit status is checked. If it is not ready, the message "TAPE XX NOT READY" is sent to the control point display and no further processing is done until the tape is ready or an error flag is set.

3. One block of data is read in odd parity. If the length is less than 4 bytes (signifying noise) it is ignored and another record is read.

4. If an end-of-file was encountered, the buffer status is changed to reflect it and an EXIT is made.

5. When a parity error is encountered, the tape is backspaced one block and reread. The message "TAPE XX PARITY ERROR" is sent if the parity error still exists after 3 attempts. A pause bit is set in RA and is cleared only after "X.GO." is typed in answer to the display message.

6. When the pause bit is cleared, the bad data is stored in the buffer and a new block is read.

7. The data is read until an end-of-record or end-of-file is sensed.

B. READ BCD TAPE

1. The requested tape unit status is checked. If it is not ready, the message "TAPE XX NOT READY" is sent to the control point display and no further processing is done until the tape is ready or an error flag is set.

2. One block of data is read in even parity. If an end-of-file was encountered, the buffer status is changed to reflect it and an EXIT is made. If the length is less than 6 bytes (signifying noise), it is ignored and another record is read.

3. If a parity error is sensed, the tape is backspaced one block and reread. The message "TAPE XX PARITY ERROR" is sent if the parity error persists after 3 attempts. A pause bit is set in RA and is cleared only after "X.GO." is typed in answer to the display message.

4. When the pause bit is cleared, the normal processing continues.

5. The number of significant BCD characters is determined and trailing spaces are suppressed by a zero byte.

6. The BCD characters are converted to display code by a table look-up. A blank (55_8) is substituted for an illegal character.

7. The data is copied into the central memory circular buffer until a zero byte is found.

8. Only one record (120 characters) is read and then an EXIT is made.

C. REWIND/UNLOAD

1. The tape is checked for ready status and if an unload was requested the tape is rewound and then unloaded.

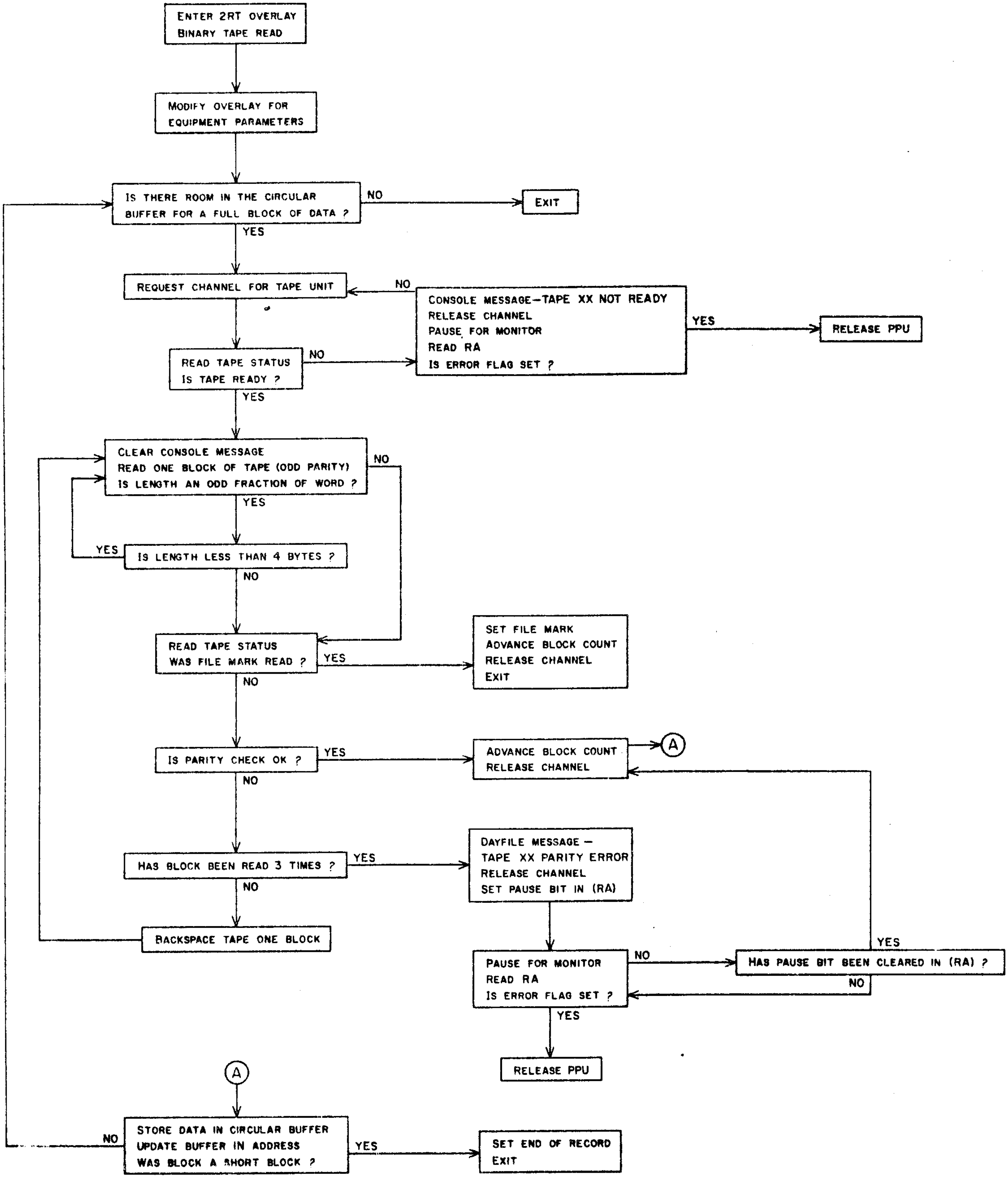
2. If only a rewind was requested, the tape is rewound.

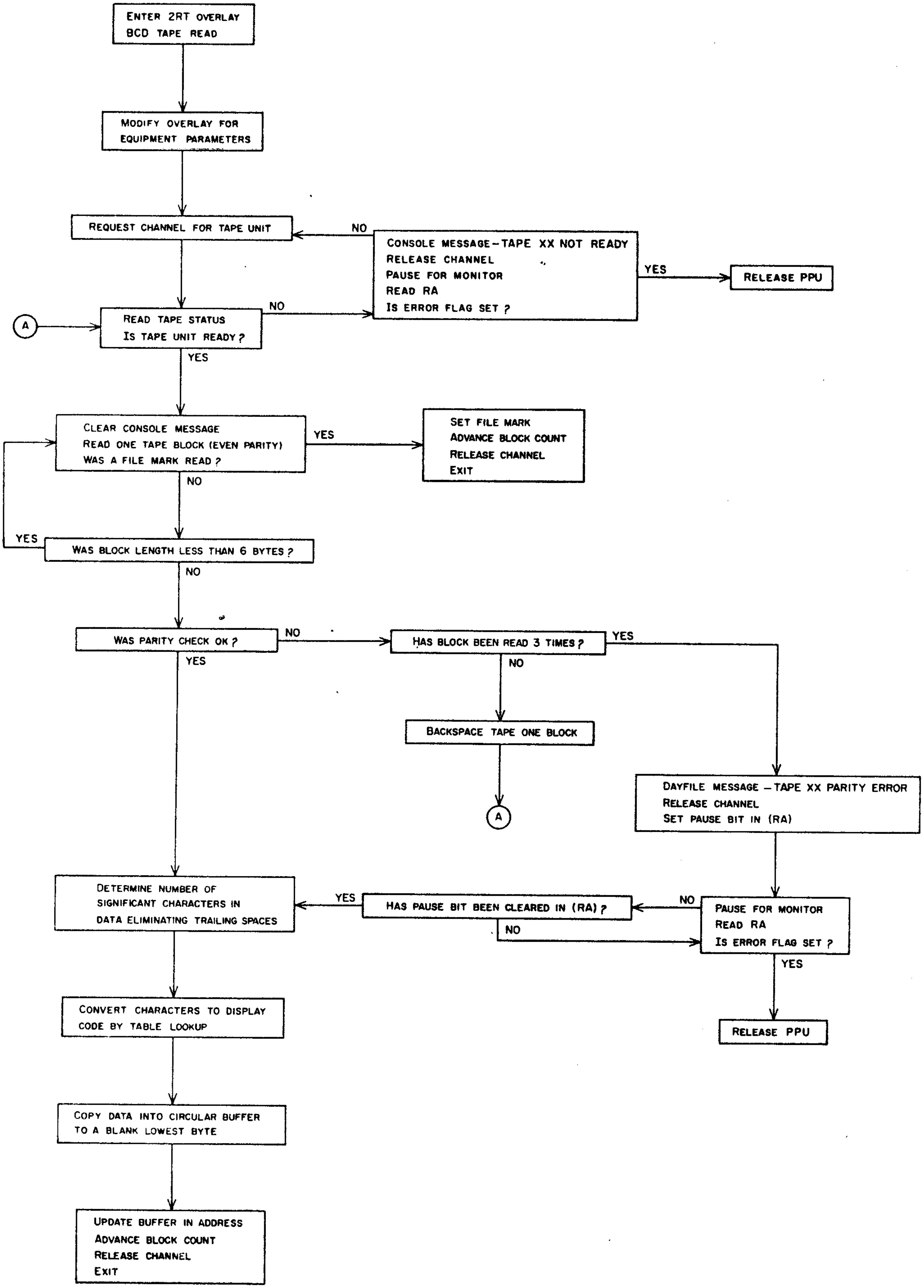
3. The block count in the FST entry is cleared and an EXIT is taken.

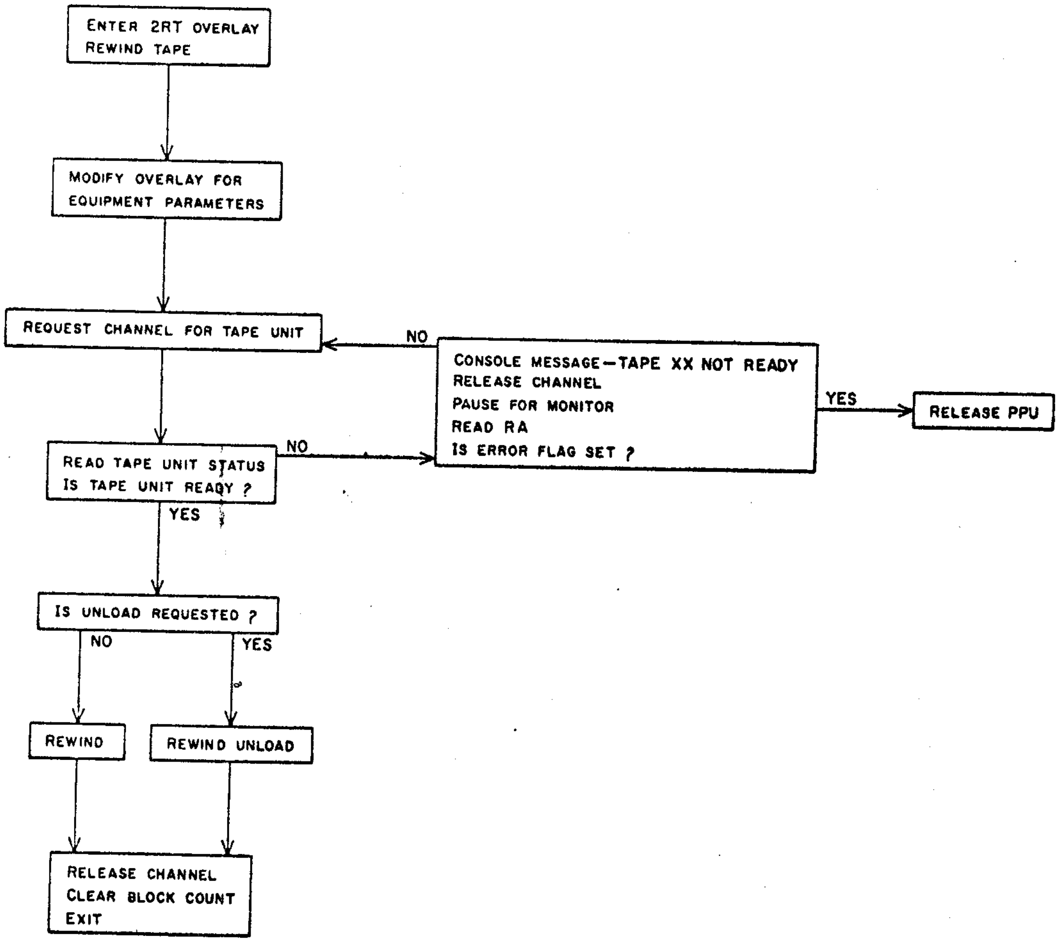
NOTES:

1. Noise records in binary is a block less than 4 bytes and in BCD less than 6 bytes.

2. BCD characters which do not have a legal display code counterpart become blanks (55_8).


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
NO
A
READ TAPE STATUS
IS TAPE UNIT READY ?
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
BACKSPACE TAPE ONE BLOCK
A
DAYFILE MESSAGE – TAPE XX PARITY ERROR
RELEASE CHANNEL
SET PAUSE BIT IN (RA)
YES
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

ROUTINE: 2TJ -- Translate Job Card

PURPOSE: To check the parameters on the job card for errors and assemble the values for use by other routines.

GENERAL: 2TJ is called by 1BJ, 1LJ, or 1LT. The job card is read from the control card buffer located in the control point area. Upper entry to 2TJ, the buffer parameters are passed through the PP's direct core cells. All job card parameters except the job name are converted from display code to binary.

METHOD:

1. If the circular buffer contains more than 95 words or 190 characters, the PP is released with a dayfile diagnostic - "TOO MANY CONTROL CARDS".

2. Otherwise, the job name is assembled in left-justified display code with trailing spaces. The job name may not be blank or begin with a number.

3. The priority is extracted and converted to binary. Only the lowest 4 bits are stored for the job priority.

4. The time limit is extracted and converted to binary. The lowest order 5 octal digits are rounded to the nearest 10_8 seconds and stored for the time limit.

5. The field length is extracted and converted to binary. The lowest order 17 bits are rounded up to the nearest 100_8 words.

6. The PPU time charges for the CP area are cleared in order to assign future PP activity to the job.

7. A dayfile message - JOB CARD ERROR - is caused by:

   a) Job name exceeding 7 characters or not beginning with an alphabetic character.

   b) Priority exceeding 7 characters.

   c) Time limit exceeding 7 characters.

   d) Field length exceeding 7 characters.

8. If any parameter is blank, a corresponding value is inserted.

   a) priority - 1

   b) time limit - 10_8 seconds

   c) field length - 40000_8 words

NOTES:

1. The routine READ NEXT CHARACTER reads one central memory word (10 characters) whenever the character string is depleted.

2. The parameters for the control statement buffer used by 2TJ, P60-65, are set by the circular buffer I/O routines.

3. 2TJ and the calling routine 1BJ, 1LJ, or 1LT are released and control reverts to the idle loop if one of the following conditions occur:

   a) Too many control cards - more than 190 characters in all control cards, excluding trailing blanks. About 40 cards can be used and this error usually occurs when a record separator (7-8-9 card) has been omitted.

   b) If the job name field is blank or absent.

   c) If the first character of the job name field is not an alphabetic character.

4. Comments should not pass column 64 because any word after that column will be interpreted as a control word.

2TJ Routines

Location	Routine	Calls
2000	Main Program	2100, 2200, 2300, 2340
2100	Assemble Argument	2140
2140	Read Next Character	-
2200	Decimal Conversion	531, 12-760
2300	Assemble Name	2100
2340	Clear PP Time	04-760

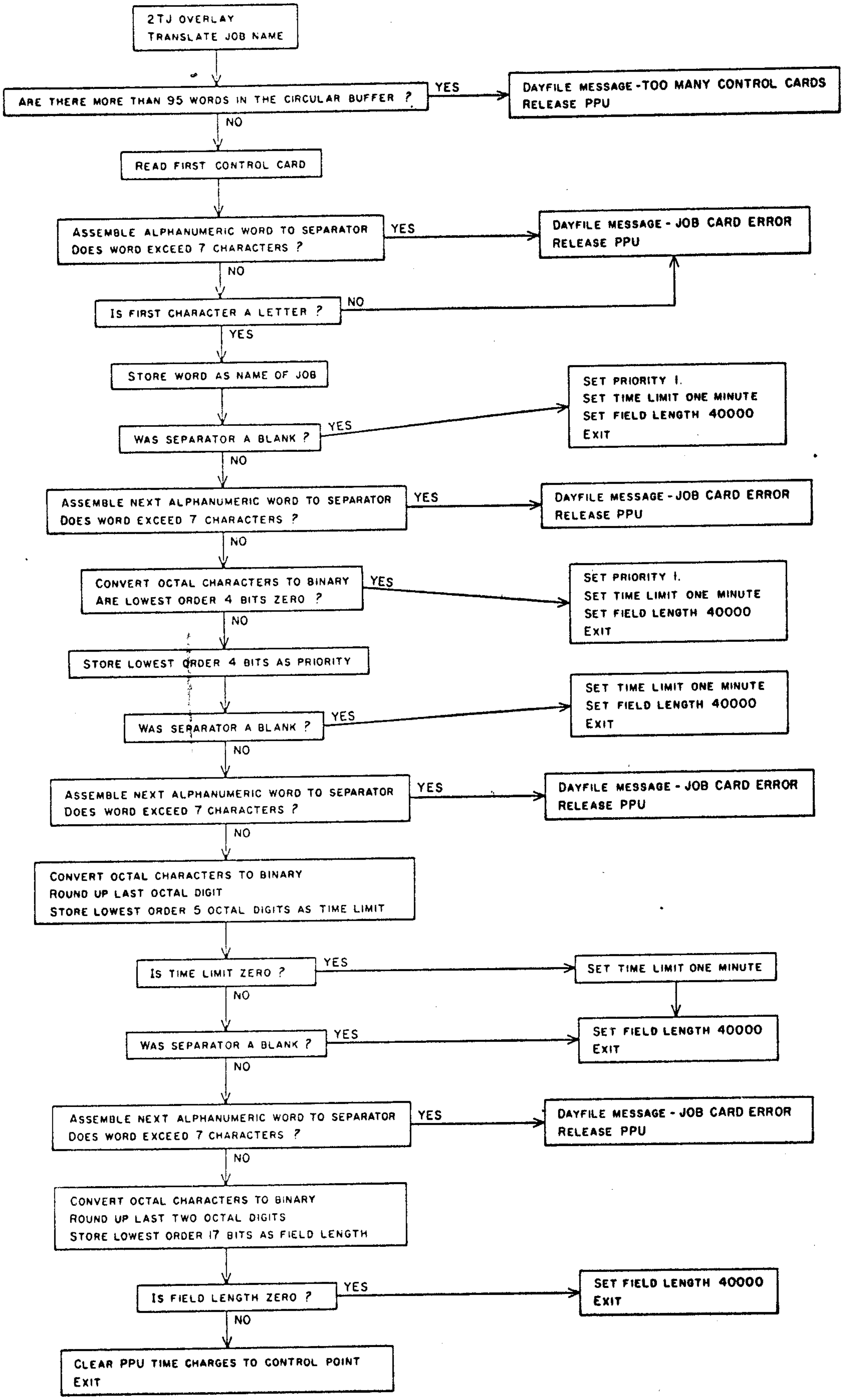
Direct Core Cells

Entry

P60/61	FIRST
P62/63	IN
P64/65	OUT
P55	RA

Exit

P30/34	Job Name
P35	Priority
P36	Time Limit (Rounded to Tens)
P37	Field Length (Rounded to Hundreds)


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
WAS SEPARATOR A BLANK ?
YES
SET PRIORITY 1.
SET TIME LIMIT ONE MINUTE
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
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
WAS SEPARATOR A BLANK ?
YES
SET TIME LIMIT ONE MINUTE
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 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

ROUTINE: 2TS -- Translate Control Statement

PURPOSE: To examine each statement in the control card buffer of the control point area and initiate the execution.

GENERAL: This package is called by 1AJ which was in turn called by MTR to advance the job status at a control point. Each time a control statement is initiated the PP is released and MTR must then reload 1AJ. This process continues until a blank entry in the control card buffer is encountered and 1AJ can continue subsequent processing.

METHOD:

1. If the next control statement is blank, all control cards have been processed so an EXIT is made to 1AJ.

2. ASSIGN

   a) No separator is required between ASSIGN and equipment type.

   b) If either field is incorrect, an error flag of 3 is set in the control point area and an EXIT made. A "CONTROL CARD ERROR" message is sent to the dayfile and the next time 1AJ is called the PP will be aborted.

   c) The file name from the card is stored and a request is made of MTR for the octal code that the equipment type designates. If a mnemonic, i.e., WT, CR, etc., instead of octal digits, i.e., 51, 42, etc., was specified, a console message "WAITING FOR XX" will appear.

   d) The file name is assigned an FNT entry with local type status. The equipment type is set into the FST entry.

   e) The control statement buffer address is advanced so that the next statement will be processed when 1AJ is reloaded.

   f) A dayfile message noting the equipment assignment (XX ASSIGNED) is sent and the PP released.

3. COMMON

   a) If the file name exceeds 7 characters, a control card error exit is made.

   b) The FNT is searched for a file name identical to the one on the card. It must be assigned to the calling control point.

   c) If the found file is not on the disk, MTR is requested to assign the proper equipment. If the request is not fulfilled, a console message "WAITING FOR XX" is sent and the PP released.

d) If there is no file assigned to the control point with the proper name, a console message "WAITING FOR COMMON FILE" is sent and the PP is released.

e) A file with the correct name and control point assignment is given common status and then the PP is released.

3. RELEASE

a) The FNT is searched for a common file with assignment to the requesting control point and a name identical to the control card. If one is found, the common type is changed to local so that when this job is logged off the file will be erased.

b) A dayfile message "RELEASE XXXX" is sent even if the file was not found.

c) A common file may be released by a job but still used by it because the file is not lost until the job is terminated.'

4. EXIT

a) If this control card is the next to be executed an exit is made from this overlay.

b) 1AJ checks the error flags before any control is given to 2TS. If such a flag is set, 2EF is called to read the rest of the control cards in the buffer and position the buffer parameters to the statement after an EXIT card, if one is found, or to a blank word, if no EXIT card was issued.

c) If no errors have thus far been encountered and an EXIT card found, 2TS will exit and 1AJ will finish the rest of its processing.

d) An EXIT card will cause job termination when encountered if no errors exist in the job.

5. REQUEST

a) If an equipment has not been assigned by the operator the message "REQUEST XXXX" is sent.

b) When the operator does make the assignment, the octal digits will appear in CP(22). This byte is cleared and a blank entry in the FNT is searched for.

c) The requested file will be given an FNT entry with local type and the equipment number will be set in the FST.

d) A dayfile message "(XX ASSIGNED)" is sent and the PP released.

6. MODE

a) The octal digit is assembled and MTR is requested to assign the corresponding exit mode.

b) A dayfile message "MODE X" is sent and the PP is released.

c) MTR will change the exit mode in the exchange package for the control point.

7. SWITCH

a) The octal digit is assembled and if it is between 1 and 6, it is stored in word 26 of the control point area. The sense switches occupy bits 6-11.

b) The sense switches are passed to the program through (RA).

c) The PP is released after a dayfile message of "SWITCH X" is sent. If the digit is not between 1 and 6, no bit is set but the dayfile message is sent.

8. PROGRAM CALL

a) FNT search

1) The FNT is searched for a file with the name identical to the one on the card and assigned to the control point. If none is found, the library is searched.

2) The file must be on disk 0.

3) The file is then read into central memory beginning at RA until an end-of-record or field length is reached.

4) The exchange area is cleared and P is then set to the number of arguments + 3 and FL is put into A_0.

5) The sense switches and lights already set in the control point area are passed to RA and RA+1 is cleared.

6) The parameters on the control card are assembled and replace their corresponding entry in the argument area. Blank parameters will cause the

original vlaue to remain. A period or closing parenthesis must terminate the parameters.

7) If the RSS (read next control statement but stop before execution) flag is set, the card is sent to the dayfile and the PP is released.

8) Otherwise, the central processor is requested of MTR to begin execution of the newly loaded program and the statement is sent to the dayfile.

b) CLD search

1) Each entry in the CLD is searched for the file name and is it is found it is read into central memory beginning at RA until an end-of-record or field length is reached.

2) The program is read in from the disk in the same manner as described under FNT search.

c) PLD search

1) If the file name is not found in FNT or CLD, PLD must contain it or an error results.

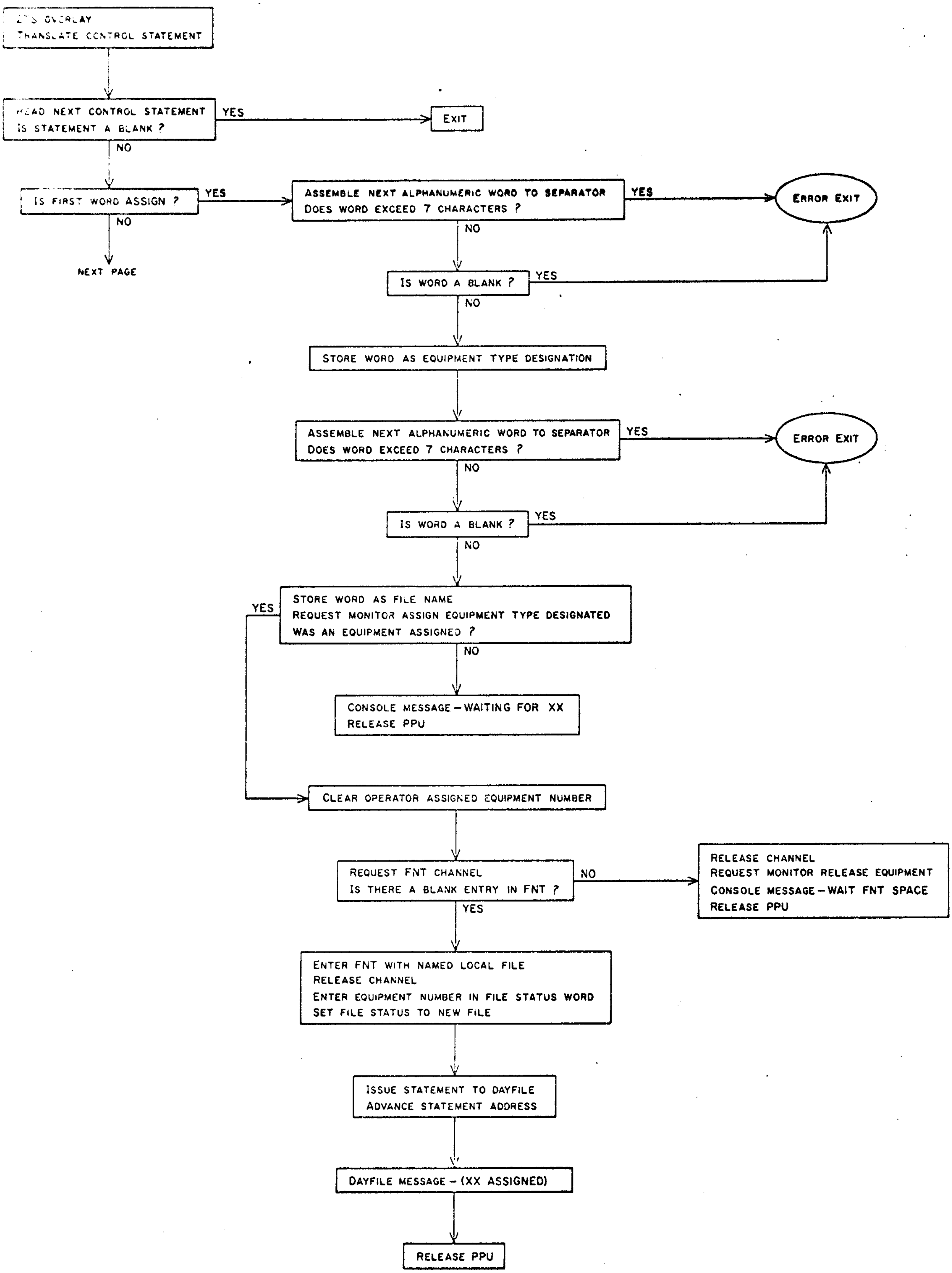
2) If the name does not begin with a letter, an error message is sent to the dayfile.

3) Parameters may appear in the call. If they do, then the first one is assembled into bits 18-35 and the second into bits 0-17 of PP recall register. If only one parameter is needed, it resides in the lower 18 bits of the register. The call is assembled in the PP recall register at the control point so that when MTR senses this request, the package will be released to a free PP.

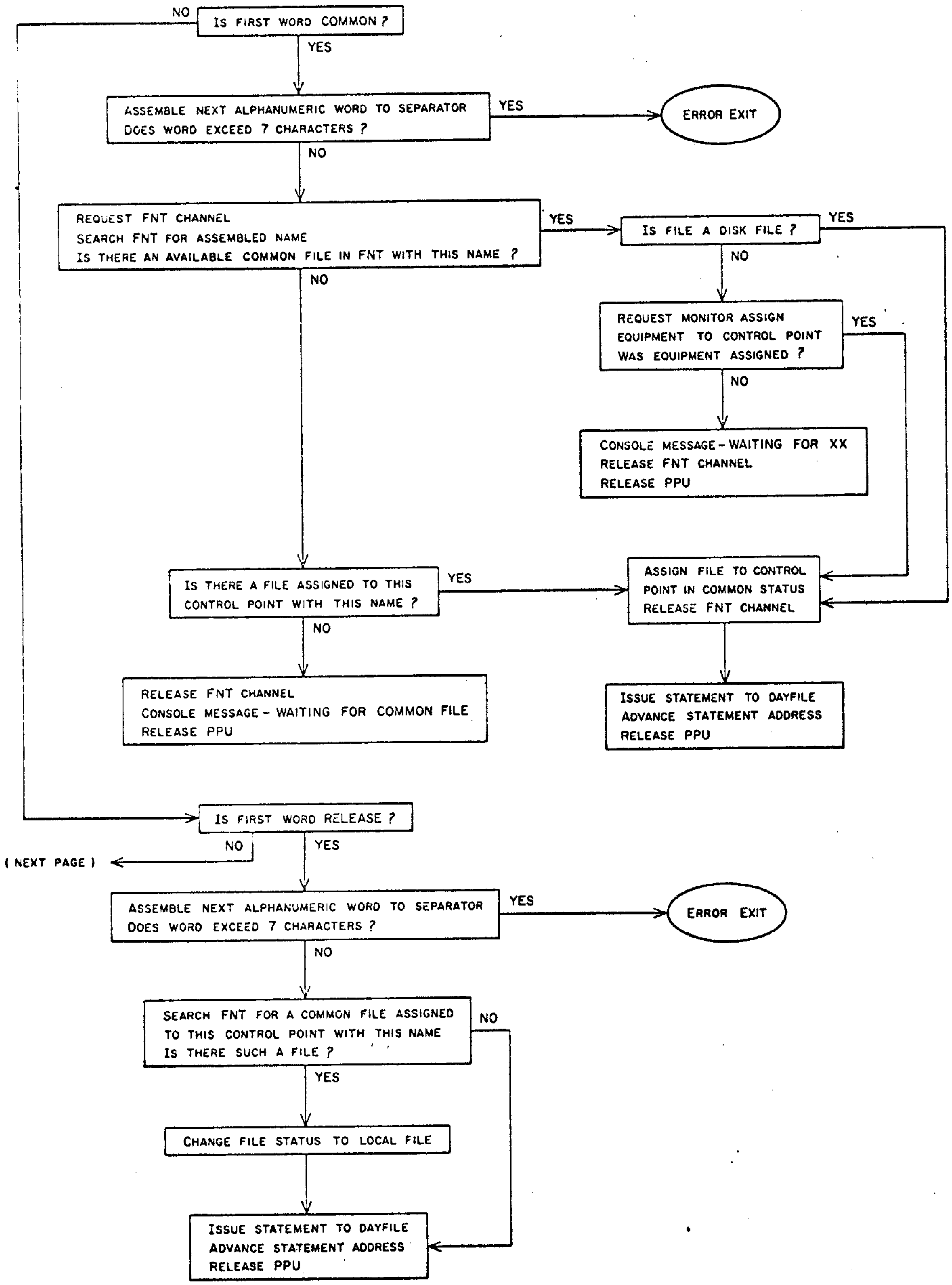
4) The statement address is advanced to the next statement and the PP is released.

2TS Routines

2000	Main Program	2100, 2200, 3200, 3300, 3540, 3100
2040	Message for ASSIGN	
2100	Unpack Next Statement	
2170	EXIT	
2200	Search for Special Format	
2240	Assemble Name	3100
2300	ASSIGN	2240, 22-760, 3360, 12-760, 2500, 2040, 3100, 3000
2400	REQUEST	2240, 2500, 2040, 12-760, 3000
2460	MODE	2240, 25-760, 3000
2500	Assign File	740, 750, 23-760, 12-760
2600	RELEASE	2240, 3000
2660	COMMON	2240, 740, 750, 3740, 12-760, 3000
3000	Issue Exit	01-760, 530, 12-760
3100	Error Exit	3000
3150	Enter Arguments in Program	2240
3200	Search for Assigned File	2240, 3400, 3460, 3150, 15-760, 3000
3300	Search CLD	2240, 3400, 3460, 3150, 15-760, 3000
3360	Console Message	
3400	Read Program	740, 700, 400, 750
3460	Clear Exchange Area	
3540	Search PLD	2240, 3700, 3000
3640	SWITCH	2240, 3000
3700	Assemble Data	
3740	Assign Equipment	22-760, 3360, 750, 12-760

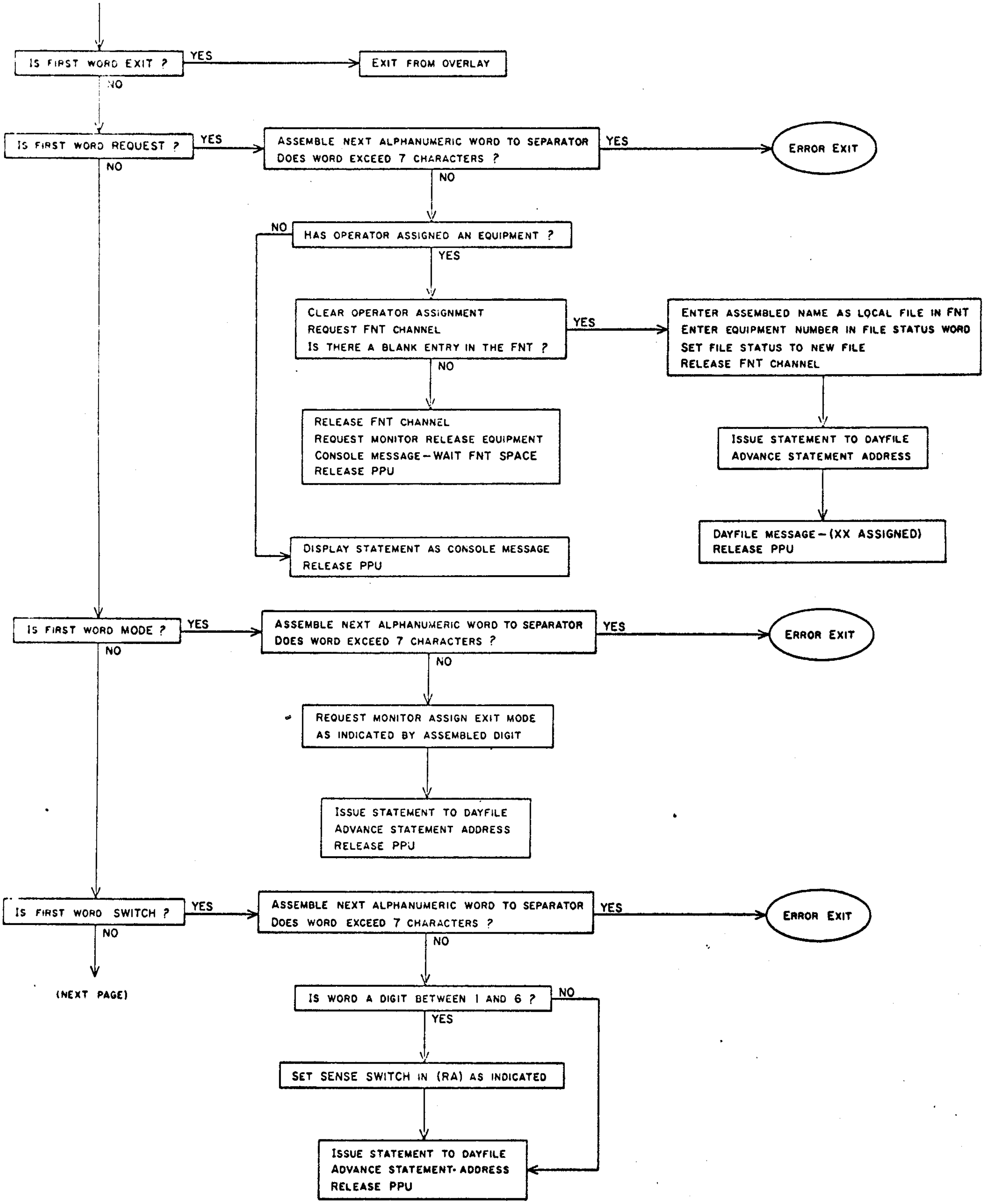

2TS OVERLAY
TRANSLATE CONTROL STATEMENT
READ NEXT CONTROL STATEMENT
IS STATEMENT A BLANK ?
YES
EXIT
NO
IS FIRST WORD ASSIGN ?
YES
NO
NEXT PAGE
ASSEMBLE NEXT ALPHANUMERIC WORD TO SEPARATOR
DOES WORD EXCEED 7 CHARACTERS ?
YES
ERROR EXIT
NO
IS WORD A BLANK ?
YES
NO
STORE WORD AS EQUIPMENT TYPE DESIGNATION
ASSEMBLE NEXT ALPHANUMERIC WORD TO SEPARATOR
DOES WORD EXCEED 7 CHARACTERS ?
YES
ERROR EXIT
NO
IS WORD A BLANK ?
YES
NO
STORE WORD AS FILE NAME
REQUEST MONITOR ASSIGN EQUIPMENT TYPE DESIGNATED
WAS AN EQUIPMENT ASSIGNED ?
YES
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

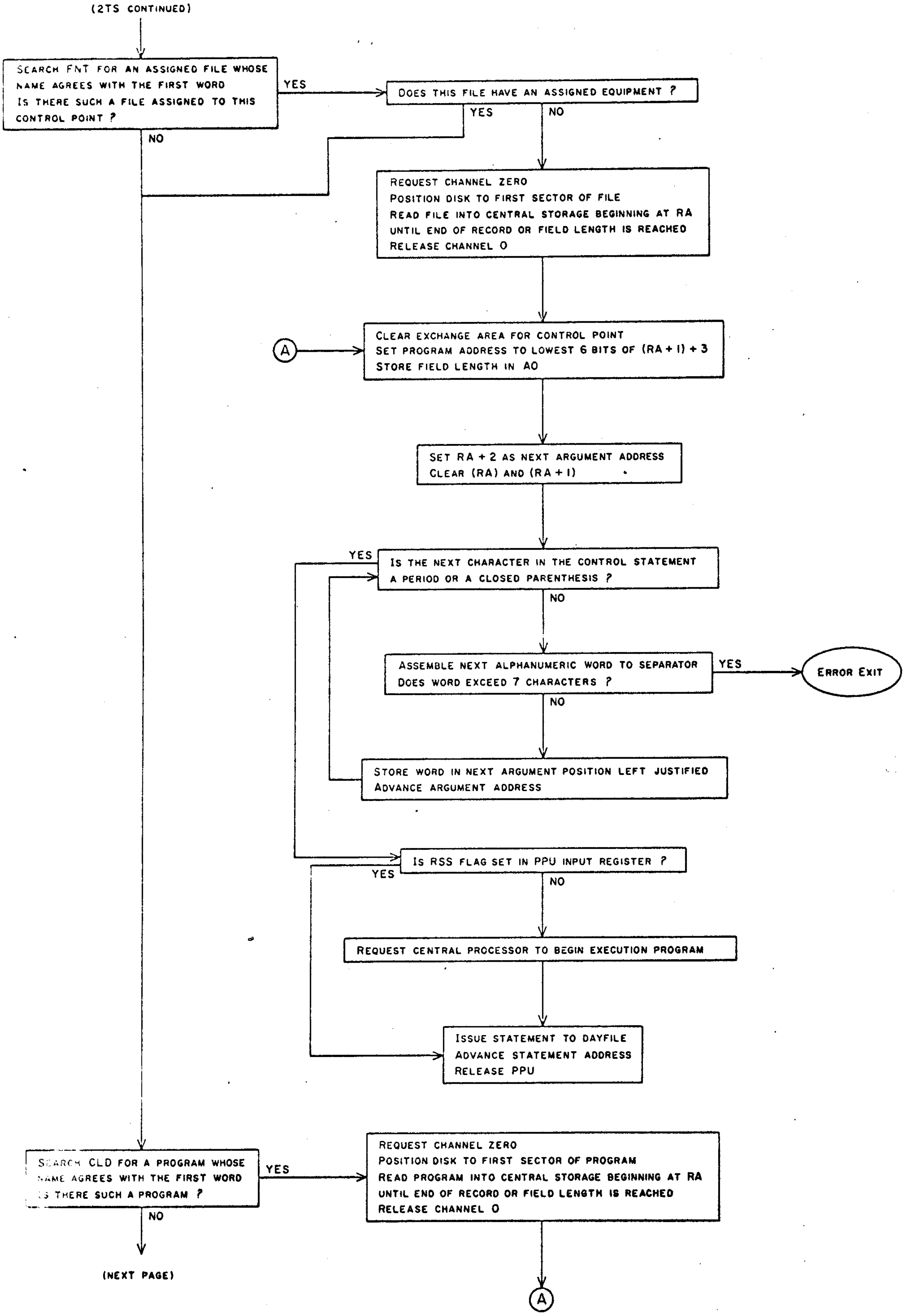
(2TS CONTINED)

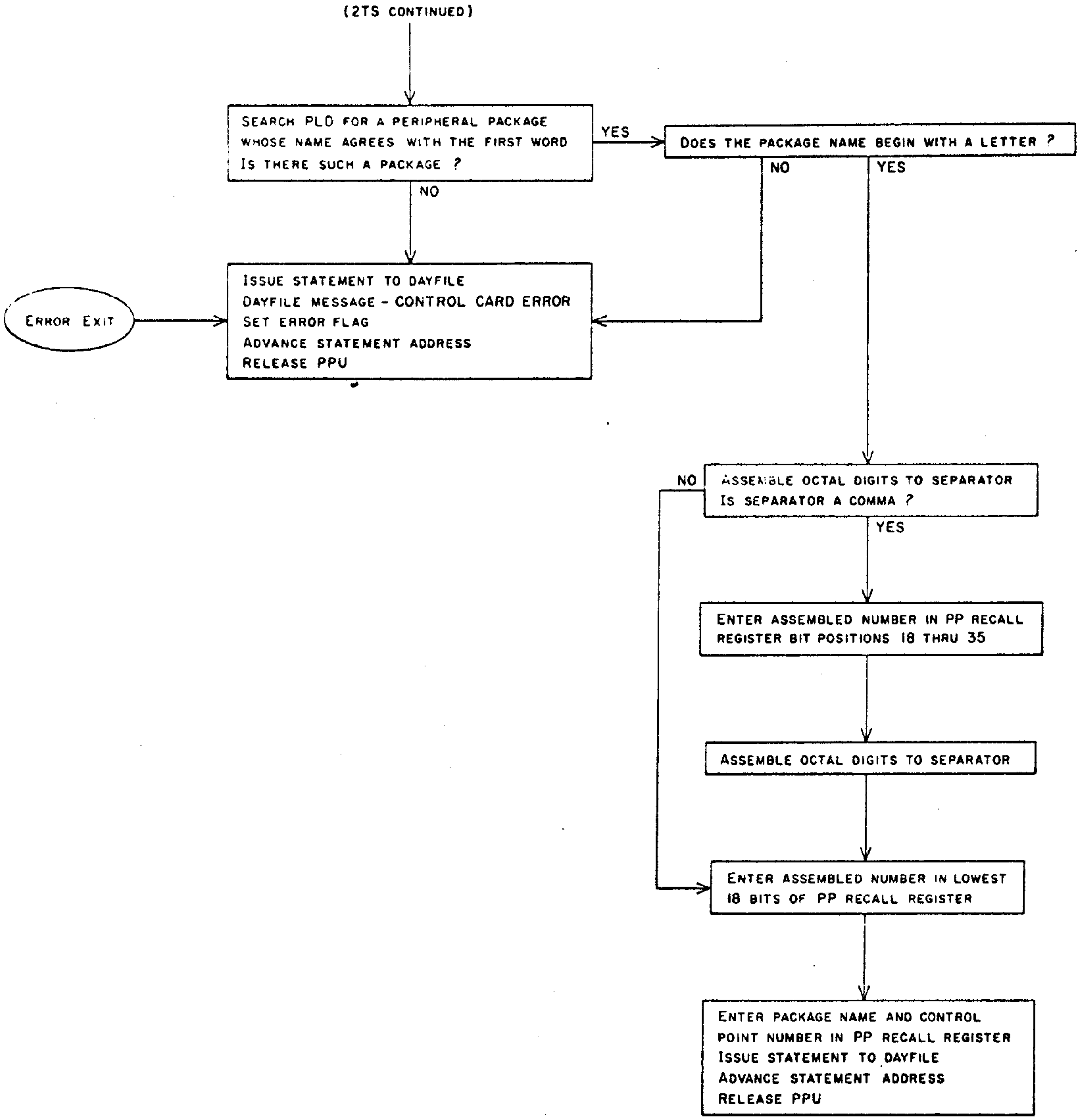


NO
IS FIRST WORD COMMON ?
YES
ASSEMBLE NEXT ALPHANUMERIC WORD TO SEPARATOR
DOES WORD EXCEED 7 CHARACTERS ?
YES
ERROR EXIT
NO
REQUEST FNT CHANNEL
SEARCH FNT FOR ASSEMBLED NAME
IS THERE AN AVAILABLE COMMON FILE IN FNT WITH THIS NAME ?
YES
IS FILE A DISK FILE ?
YES
NO
REQUEST MONITOR ASSIGN
EQUIPMENT TO CONTROL POINT
WAS EQUIPMENT ASSIGNED ?
YES
NO
CONSOLE MESSAGE - WAITING FOR XX
RELEASE FNT CHANNEL
RELEASE PPU
NO
IS THERE A FILE ASSIGNED TO THIS
CONTROL POINT WITH THIS NAME ?
YES
ASSIGN FILE TO CONTROL
POINT IN COMMON STATUS
RELEASE FNT CHANNEL
NO
RELEASE FNT CHANNEL
CONSOLE MESSAGE - WAITING FOR COMMON FILE
RELEASE PPU
ISSUE STATEMENT TO DAYFILE
ADVANCE STATEMENT ADDRESS
RELEASE PPU
IS FIRST WORD RELEASE ?
NO
YES
(NEXT PAGE)
ASSEMBLE NEXT ALPHANUMERIC WORD TO SEPARATOR
DOES WORD EXCEED 7 CHARACTERS ?
YES
ERROR EXIT
NO
SEARCH FNT FOR A COMMON FILE ASSIGNED
TO THIS CONTROL POINT WITH THIS NAME
IS THERE SUCH A FILE ?
NO
YES
CHANGE FILE STATUS TO LOCAL FILE
ISSUE STATEMENT TO DAYFILE
ADVANCE STATEMENT ADDRESS
RELEASE PPU

(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
NO
IS FIRST WORD MODE ?
YES
ASSEMBLE NEXT ALPHANUMERIC WORD TO SEPARATOR
DOES WORD EXCEED 7 CHARACTERS ?
YES
ERROR EXIT
NO
REQUEST MONITOR ASSIGN EXIT MODE
AS INDICATED BY ASSEMBLED DIGIT
ISSUE STATEMENT TO DAYFILE
ADVANCE STATEMENT ADDRESS
RELEASE PPU
NO
IS FIRST WORD SWITCH ?
YES
ASSEMBLE NEXT ALPHANUMERIC WORD TO SEPARATOR
DOES WORD EXCEED 7 CHARACTERS ?
YES
ERROR EXIT
NO
(NEXT PAGE)
NO
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
ERROR EXIT
ISSUE STATEMENT TO DAYFILE
DAYFILE MESSAGE - CONTROL CARD ERROR
SET ERROR FLAG
ADVANCE STATEMENT ADDRESS
RELEASE PPU
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

ROUTINE: 2WT -- Write Tape

PURPOSE: To write both binary and BCD blocks of data on magnetic tape.

GENERAL: Once a write code is detected in the request parameter, a call is made to the CIO Write Function routine which then checks the equipment type of the file. When a file type of tape is determined, a call is made to load 2WT. When the mode of binary or BCD is determined, the appropriate transfer is made by CIO.

METHOD: A. BINARY WRITE

1. The circular buffer is checked to determine if there is a full block of data. If there is not, and an end-of-record function is not requested, execution returns to the CIO Write Function routine.

2. If end-of-record is requested, the last partial record will be written.

3. A transfer is made to subroutine Write Binary Tape in 2WT to actually write the block. A check is made for tape ready. If the tape is not ready, a message is displayed and a pause is executed waiting for tape to be made ready or the error flag set in RA.

4. Once tape is ready, the data is written and a parity check is made. If there is parity, a message is displayed, the tape is backspaced, and rewritten until either the parity does not exist or the error flag has been set in RA by monitor.

5. If a good write is performed, the OUT address of the buffer is then updated. If a short block was written meaning end-of-buffer, IN and OUT are set equal to FIRST and EXIT is taken to CIO Write Function.

6. If the buffer is not empty, more data is written until a short record is encountered.

B. BCD WRITE

1. If the request is a BCD write request, a jump is made from CIO Write Function to the subroutine WRITE BCD TAPE at location 2640.

2. A check is made to see if there is data in the buffer. If there is none, and the end-of-record is requested, IN and OUT are set equal FIRST. If end-of-record is not requested, an EXIT is taken.

3. If the buffer is not empty, one word at a time is read from the buffer and it is converted from display code to BCD advancing OUT as each word is read. Whenever the last byte of a word is zero, the line is padded with spaces up to 120 characters.

4. When a full line of data is made up, a jump to 3001 is taken (WRITE CODED RECORD) to write the record. The same write and parity checking operation is done here as in the binary write.

5. When a good write is completed the block count is advanced, the channel released, and more data is written until the buffer is empty.

C. WRITE FILE MARK

1. If a file mark is requested, a jump is taken from CIO Write Function to WRITE FILE MARK.

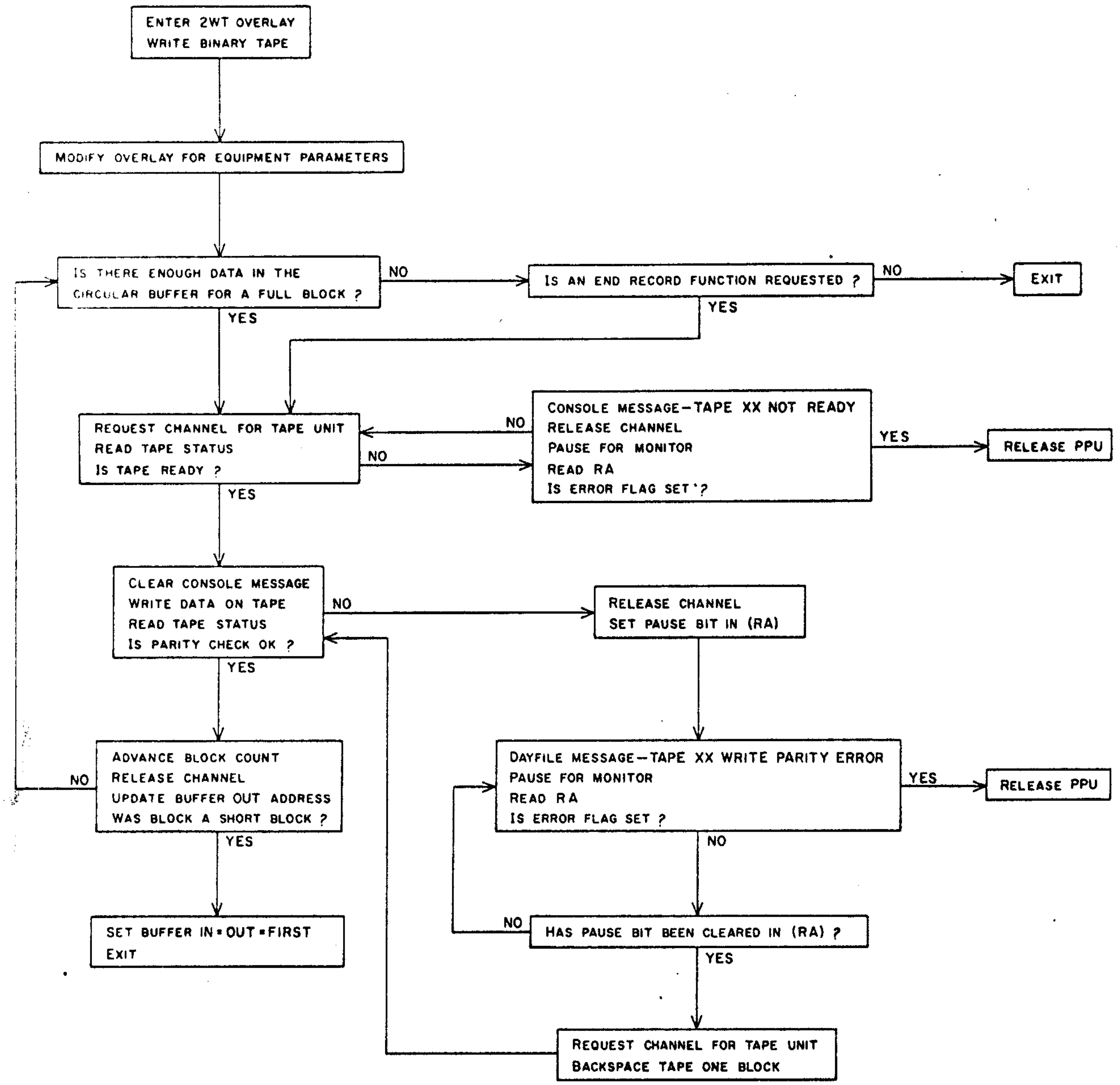
2. This routine simply finds the tape, makes sure it is ready, writes a file mark, advances the block count, and releases the channel.

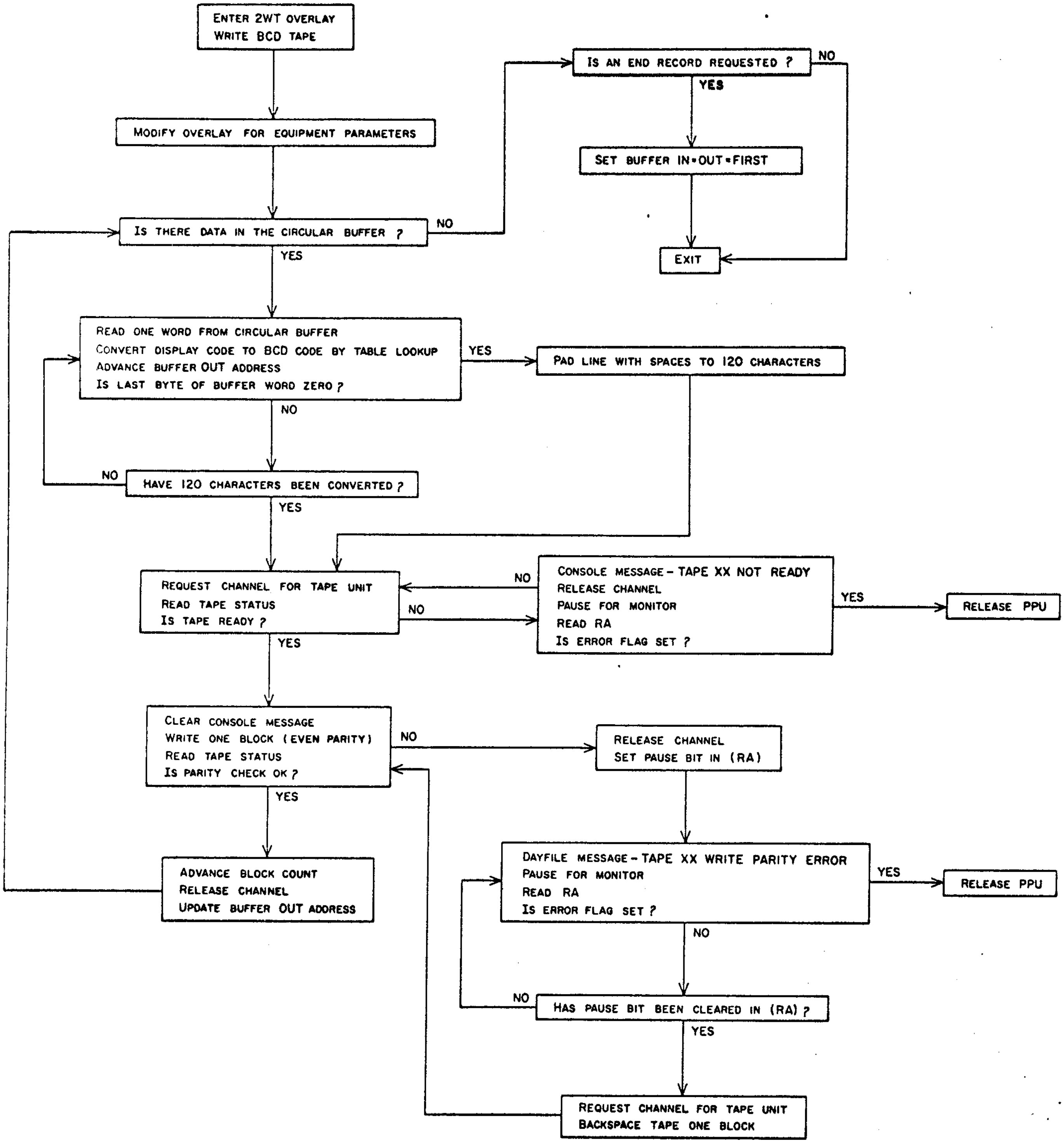
NOTES:

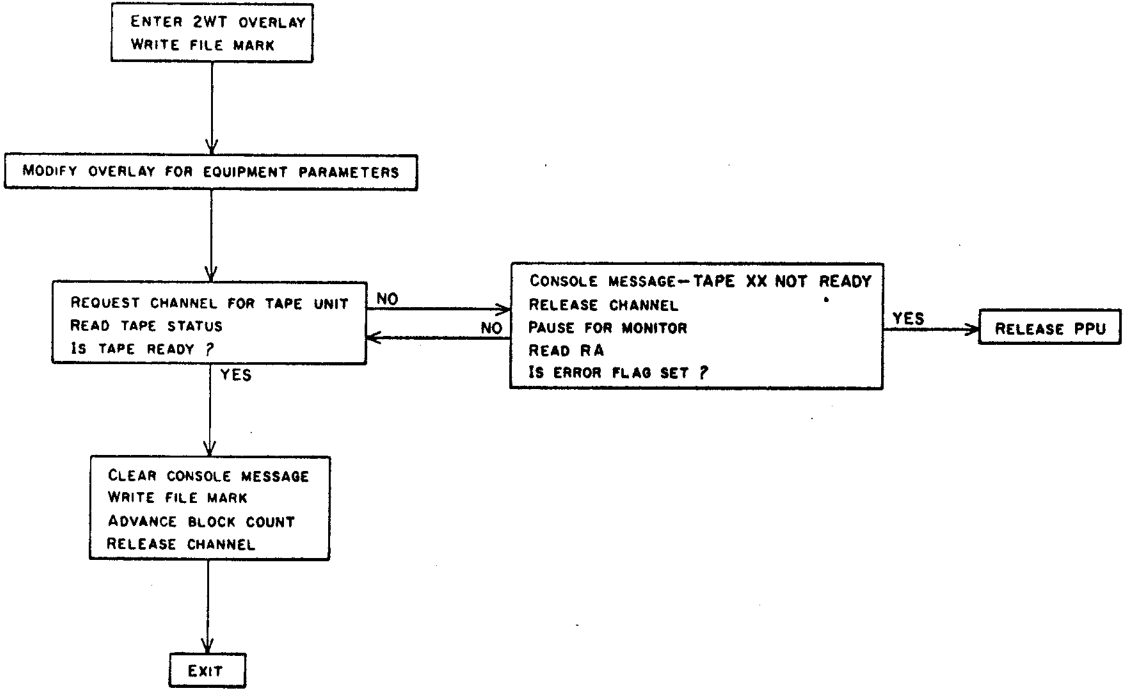
1. An end-or-record write must be issued to empty a buffer which does not contain a full block of data.

2. Binary tape records has a maximum size of 1000_8 central memory words.

3. BCD tape records are all 120 characters (one print line). Each record is padded with spaces to maintain the proper size.


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
SET BUFFER IN=OUT=FIRST
EXIT
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 BCD TAPE
MODIFY OVERLAY FOR EQUIPMENT PARAMETERS
IS THERE DATA IN THE CIRCULAR BUFFER ?
NO
IS AN END RECORD REQUESTED ?
NO
YES
SET BUFFER IN=OUT=FIRST
EXIT
YES
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
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 FILE MARK
ADVANCE BLOCK COUNT
RELEASE CHANNEL
EXIT

CONTROL DATA CORPORATION

Development Division - Applications

CIRCULAR INPUT OUTPUT

Chippewa Operating System

10/20/65

CIRCULAR INPUT OUTPUT

CIO

INTRODUCTION

All input and output for a file is passed through a circular central memory buffer. Buffer parameters are initialized by the central memory program and then the CIO package is called to perform the transfer to or from the physical medium of the file. These parameters are altered by CIO or the central program as data is inserted or extracted from the buffer. A circular effect is achieved by allowing the data to wraparound the buffer whenever the limit address of the buffer is reached. For example, on an input request data is inserted into contiguous words until the last address of the buffer is encountered. The next piece of data will be stored in the beginning address of the buffer so that the total capacity of the buffer may be utilized. All system central memory buffers, i.e. dayfile, etc., use this circular motion even if CIO is not specifically called to perform the I/O operation.

CALLING SEQUENCE

A program requesting I/O must set up certain buffer parameters. The location of these parameters is sent to CIO via the lower 18 bits of RA+1. These parameters, along with the buffer itself must reside within the field length of the job, and their addresses are relative to RA.

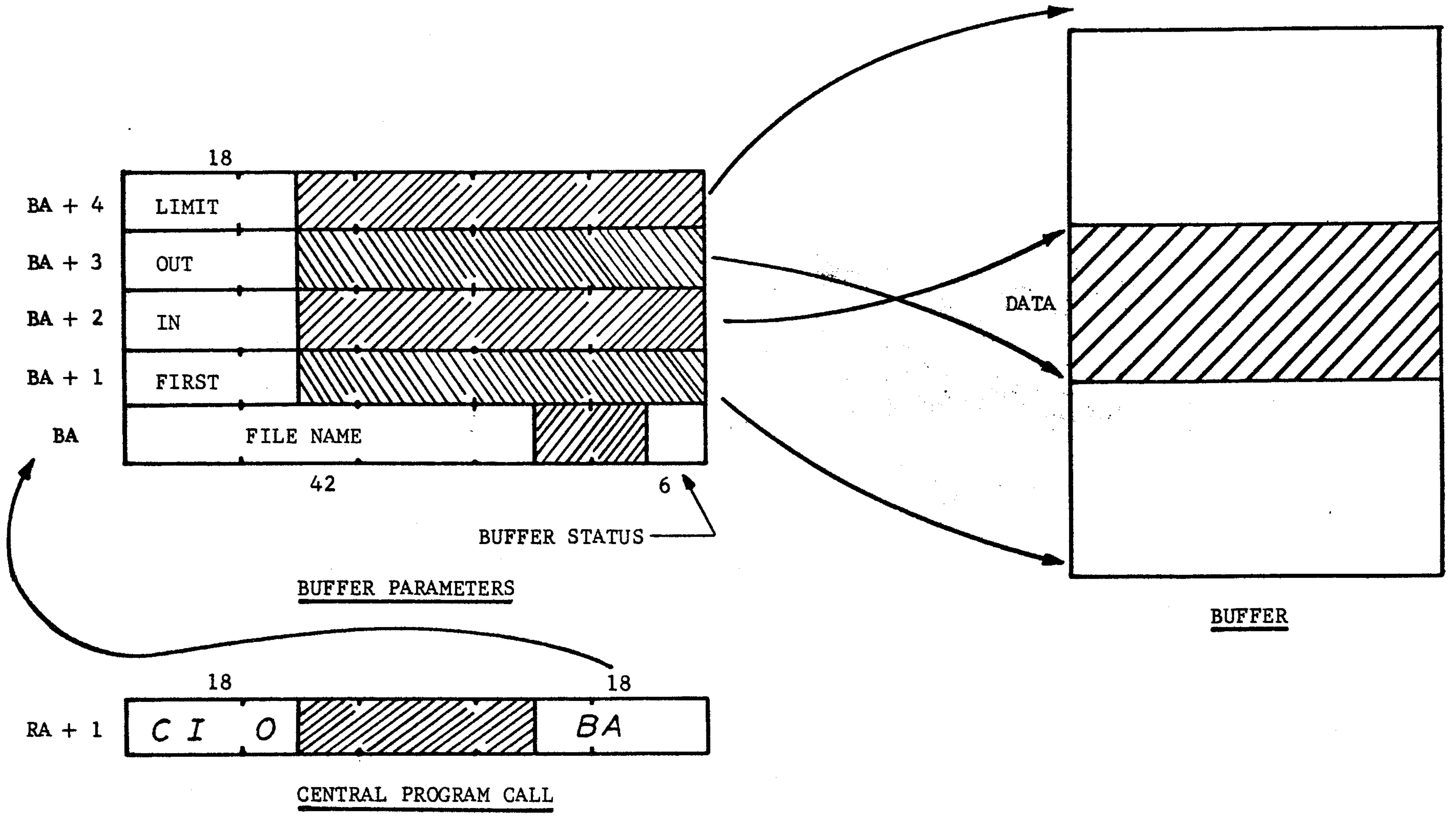
Five central memory words, designated as BA to BA+4, hold the parameters. In the first word is the name of the file in left-justified display code to be acted upon and a six bit code called the buffer status. The first digit of the buffer status specifies the type of operation: the second gives the direction (read/write) and the mode (coded/binary).

BA+1 contains the beginning address of the buffer and is called FIRST. Along with LIMIT, the last address of the buffer plus one, FIRST remains dormant, i.e. CIO never changes these values. No data is stored in LIMIT. When LIMIT is reached, the next available address for storage is FIRST. The buffer capacity is referred to as the area between FIRST and LIMIT-1.

The two remaining words, BA+2 and BA+3, are the actual pointer addresses. IN (BA+2) defines the next available address for insertion of data into the buffer. OUT (BA+3) holds the address for removal of data from the buffer. Therefore, the amount of data residing in the buffer is that between IN and OUT. IN is advanced around the buffer, but never passing OUT so as not to overstep the buffer capacity, by a 'read' operation. Any 'write' request causes OUT to move in the direction of IN and to pass data from the buffer to the file in its advance.

Either CIO or the central program may update IN and OUT. By moving IN, CIO could read data from a file to the buffer and the central program could remove the data from the buffer for its own use by moving OUT. The opposite effect would result if the central program inserted data into the buffer by incrementing IN and CIO transferred the buffer data to the file by moving OUT.

Initially, the buffer parameters are set FIRST ∓ IN =OUT with IN and OUT circling the buffer as data is inserted or removed. An empty buffer is reflected by IN = OUT. This condition is distinguished from a full buffer, IN = OUT-1, by an unused word between IN and OUT. The useable data in the buffer begins at OUT and continues (circling the buffer if necessary) to IN-1.



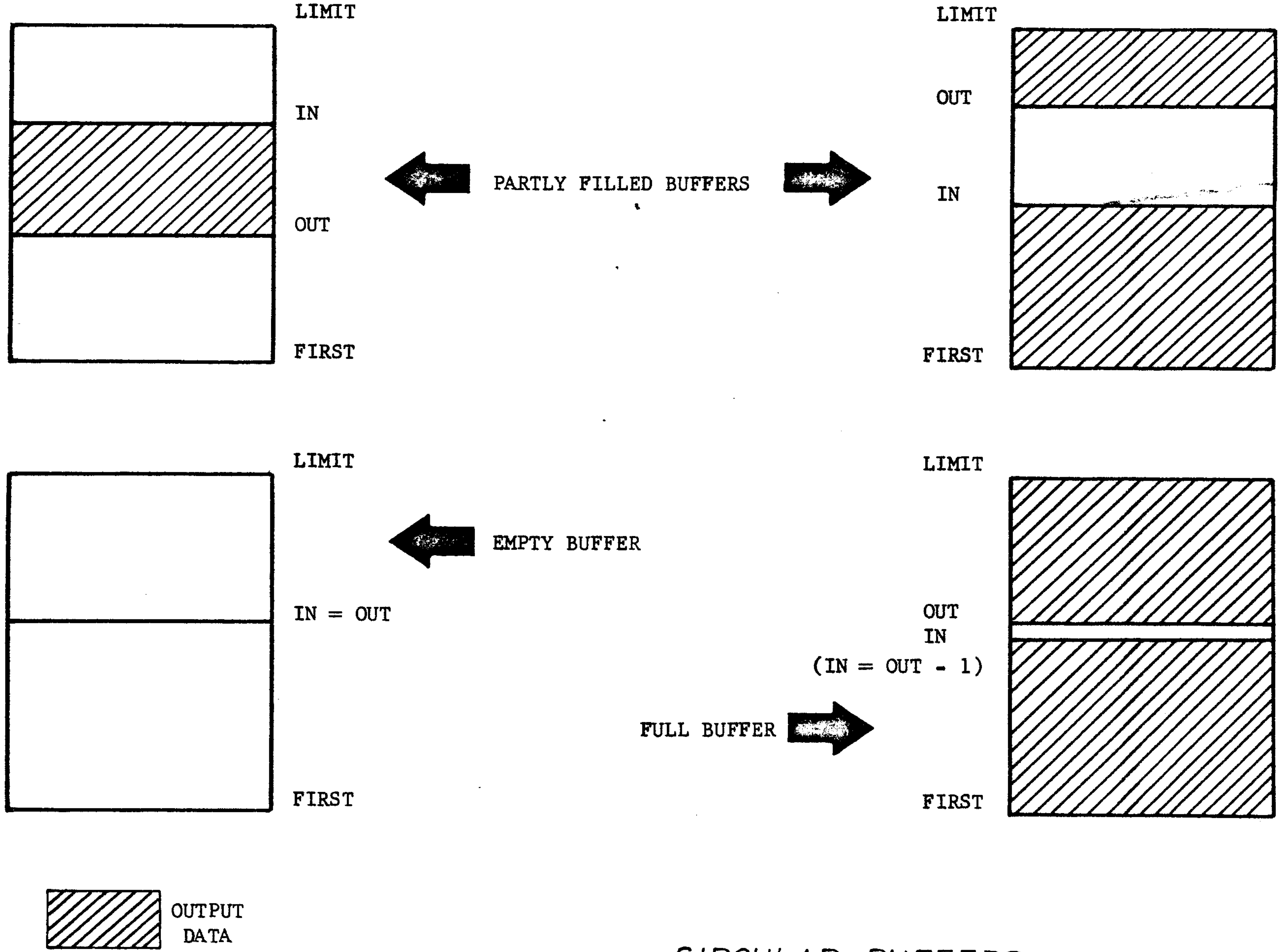
CIO PARAMETERS

BUFFER STATUS

The buffer status appears as a 2 digit octal code in the lower 6 bits of BA. This code indicates the mode of the buffer and provides an interlock for peripheral package activity. The buffer status has an even value when CIO is called. It is set to an odd value when the peripheral package has completed the I/O function. This six bit code is also kept as the last buffer status in the FST entry for the file. Whenever this value is checked and found to be even, the file is assumed to have an operation being performed on it, i.e. it is active. An odd value means that the file is not busy and is available for use.

Normal reads and writes use 'buffer I/O' as the type of operation (first octal digit with the second digit specifying the mode). This is interpreted by CIO as a request to transfer as many records as possible between the file and the buffer. A short record, i.e. end-of-file or end-of-record, or a full buffer will terminate a read operation and a write request is stopped whenever OUT = IN. If a read was requested, CIO will alter the code to indicate whether an 'end-of-record' or 'end-of-file' was read. Whenever the buffer is to be emptied to the file by a write operation, the central program must issue either an 'end-of-record' or 'end-of-file' write. This causes all of the data to be transferred and an end-of-logical-record' or 'end-of-file' to be written on the file. Therefore, if the buffer does not contain a full record of data and an 'end record' write was not issued, no data will be transferred.

When MTR accepts the I/O request by assigning CIO to a PP, RA+1 is cleared. In order for the central program to know when the PP has finished the I/O operation, the buffer status must be checked for an odd value.



CIRCULAR BUFFERS

If a 'read' was requested, the first octal digit may have been altered by CIO, but otherwise, only the second digit is incremented by one. On a 'binary backspace' the first digit is a 4 and the second may be either a 2 or 6 because only the second bit in the code is checked for the set condition.

INTERNAL STRUCTURE

Whenever an I/O operation is to be performed on a file, the CIO package is assigned to a peripheral processor. CIO examines the request and passes control within itself to the proper function. The buffer parameters are checked for legality to insure that the operation remains within the job's field definition.

A file may be of any equipment type - disk, card reader, card punch, line printer, or magnetic tape. The driver for each operation on a piece of equipment is written as a separate routine. The CIO function decides which driver is needed and calls it into the PP as an overlay. These overlays do the physical I/O and update the buffer parameters accordingly. Whenever their task is finished, CIO completes the request so that the calling program may continue its execution.

OPERATION

Each function (read, write, or backspace) within CIO calls special overlays. These overlays do more specific parameter checking to insure that the buffer can contain the amount of data requested. Parity error checking and buffer status updating are also the responsibilities of the overlay.

The Read function calls 2RD (read disk), 2RT (binary tape read), 2RC (read cards), and 2RT (BCD tape read). IN is incremented to reflect the number of words read from the file to the buffer and the first octal digit of the buffer status may be changed if an 'end-record' or 'end-file' is read.

Data is read by 2RD one sector (100_8 central memory words) at a time until a short sector is encountered or the buffer is filled. If a disk parity error is found, the sector is reread with varying margins three times and if it persists the PP stops. Only dead start will force reinitialization.

2RC transfers only useable data to the buffer by suppressing trailing blanks with a zero byte (12 bits). Ten characters per word are translated from Hollerith to display code and packed until a zero byte is inserted to signal 'end-of-physical-record'. A 7-8-9 card causes a short sector to be transferred. Only one file mark may appear within one diskfile, so when a 6-7-8-9 card is found, an 'end-of-record' sector is copied along with a second short sector to indicate end-of-file.

A binary record of less then 4 bytes is considered a noise record by 2RT. If this overlay discovers that there is not enough room in the buffer to handle a full block of 512 words, no data is transferred. A read is tried 3 times before a parity error message is sent to the dayfile. Only one block of data is read per request. Rewinding is also done by this overlay.

A BCD tape record is a constant 120 characters long. All trailing spaces are eliminated by a zero byte and the BCD characters are translated into display code. A record less than 6 bytes is considered noise and a record is read 3 times before a parity error message is sent.

The Write function is in-charge-of updating OUT. As data is removed from the buffer and copied to the file, OUT moves in the direction of IN until OUT = IN. Only a short record request will cause the buffer to be completely dumped of information and the appropriate indicator to be written on the file.

A check is made to see if the last reference to a disk file was a write. If it was not, the tracks thus far reserved by the file are dropped by 2DT. This provides multi-use of a file. Data written on a file can be backspaced and read and another write request will cause the beginning of the file to be referenced.

2WD is loaded to write disk data. If there is not enough data in the buffer for a full sector (64 words) and an 'end-record' was not requested, no data is written on the file. Every write is terminated by an EOF sector but since none of the parameters are advanced, the next write request will write over this sector. It prevents a file from ever running away. Two tracks are requested at once so that time is not wasted whenever one track is filled and another is needed.

To punch both binary and Hollerith cards, 2PC is loaded. This overlay is called whenever a file has been assigned to the card punch by a control card and a write operation requested on the file. Eighty characters or the number of characters to the first zero byte are assembled from display code to Hollerith. A 7-8-9 or 6-7-8-9 card is punched if requested. In the case of a binary request, 15 words of data are punched on a card with the appropriate binary controls - word count, 7-9 punches in column one, checksum, and sequence number.

2LP is loaded to print the file assigned to a line printer. A print line consists of either 130 characters or the number of characters to a zero byte. Page spacing is checked by this overlay.

To write a binary tape 2WT is loaded. This overlay is called into play to do all writes on 1" tape and binary writes on ½" tape. Coded records on 1" tape are in packed display code and terminated by a zero byte. A logical record consists of 1000_8 central memory words. If a parity error is encountered, the tape is backspaced and rewritten with no erasing until a good write is made or an error flag is set. 2WT also writes a file mark when one is requested.

2WT is loaded to write BCD tape. All BCD tape records are 120 characters long. If a zero byte is found before 120 characters have been converted from display code to BCD, the record is padded with spaces until 120 characters are reached. The writing continues for full blocks of data contained in the buffer until an 'end-record' or 'end-file' is requested to empty the buffer.

The Backspace function is called to backspace either binary or coded records. An end-of-file is considered a record or a coded line in each mode respectively. This action causes IN to be advanced down the buffer and a read is not necessary after a backspace to make the data available for use. A binary disk backspace may be very slow. Since a record can be written on several tracks, each pointer word before each sector must be checked for a track change. If a file contains only one record, a rewind operation is much faster.

2BD does either binary or coded backspacing on the disk. A binary backspace is done until a short sector is found. The file will be positioned either in front of the file mark just written or at the beginning of the last record. Only one coded line is backspaced with this request. OUT will reflect the address before the last card image or zero byte. No read is required to bring the data back in because the pointer words are properly adjusted.

2BT is loaded to perform the same backspace operations on tape. The physical tape is moved.

RECALL

The central program retains control of the central processor while CIO is performin the I/O operation. MTR clears RA+1 when the CIO request is accepted informing the central program to continue processing. If no further processing can be done until the data is transferred, the central processor should be given to another job. By inserting an RCL call (recall) in RA+1, control is taken away from the central program by MTR and switched to another job. Control is regained when a PP completing an operation tells monitor to recall the proper central program, or a time span of near 250 ms. has lapsed. Effective use of recall allows the central processor to be utilized more efficiently.

A workable sequence of events that will allow the central processor to execute other jobs while an I/O operation is holding up a central program is:

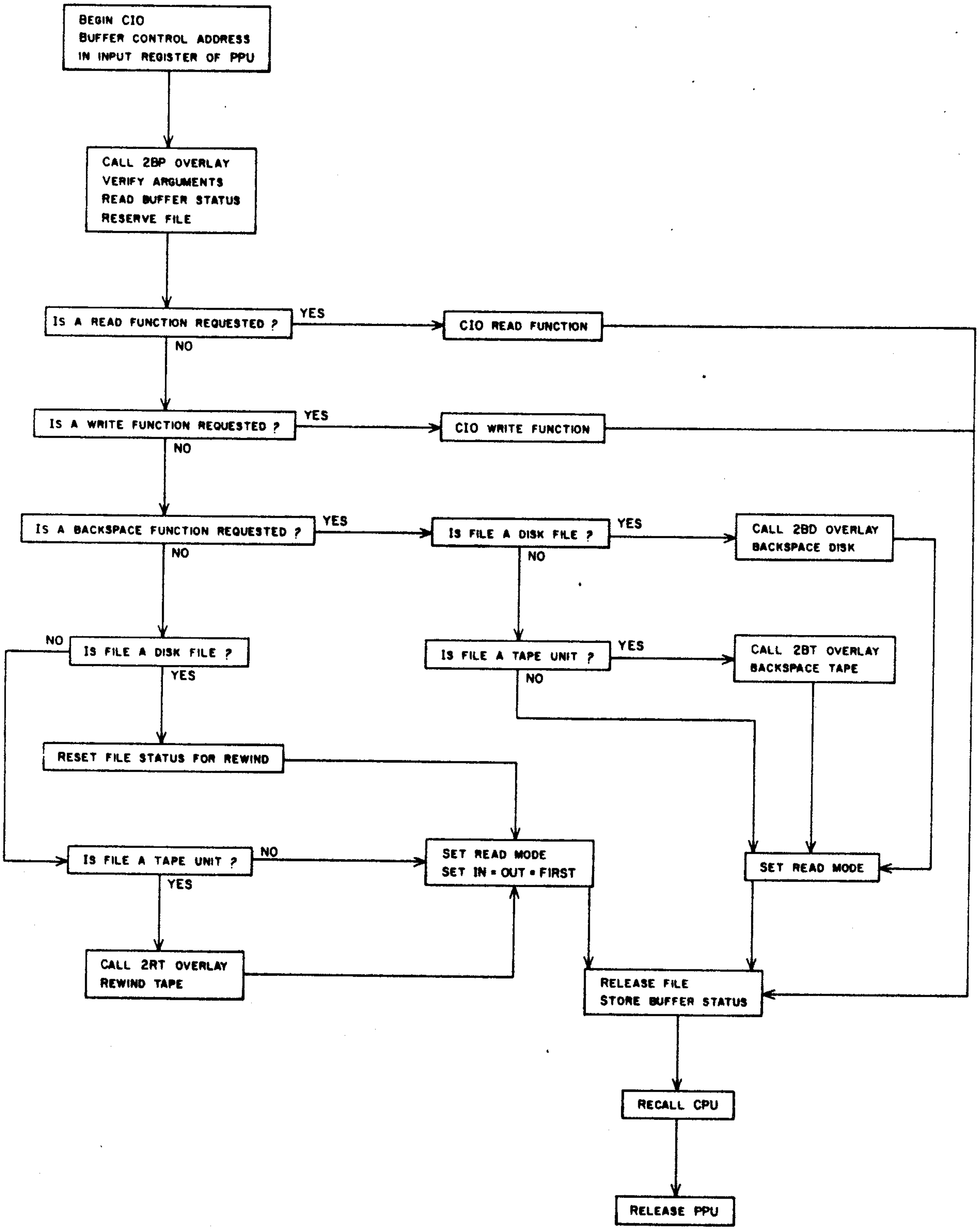
1) Send CIO call to RA+1
2) Wait until MTR has accepted the request by clearing RA+1
3) Check buffer status for an odd value.
4) If an odd value is found, continue normal processing, otherwise send RCL call to RA+1
5) Repeat steps 2-4, exiting only if the buffer status is odd

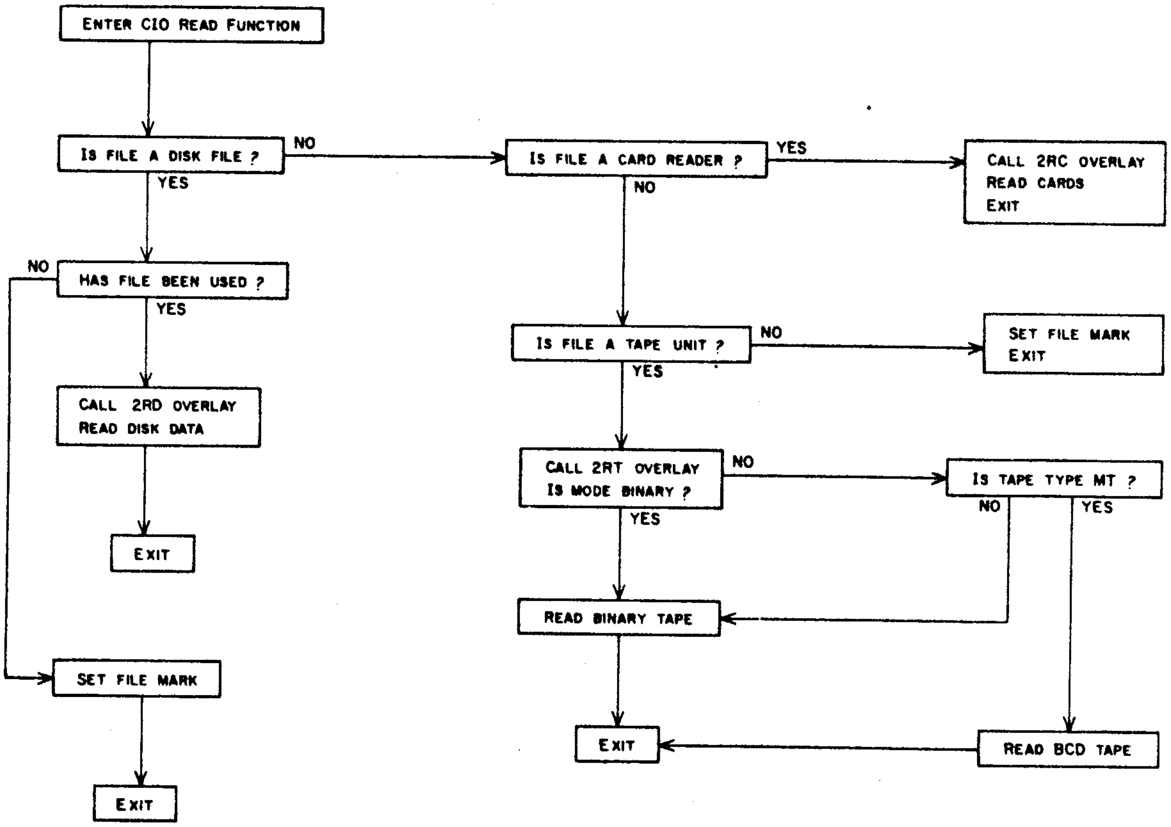
BUFFER STATUS

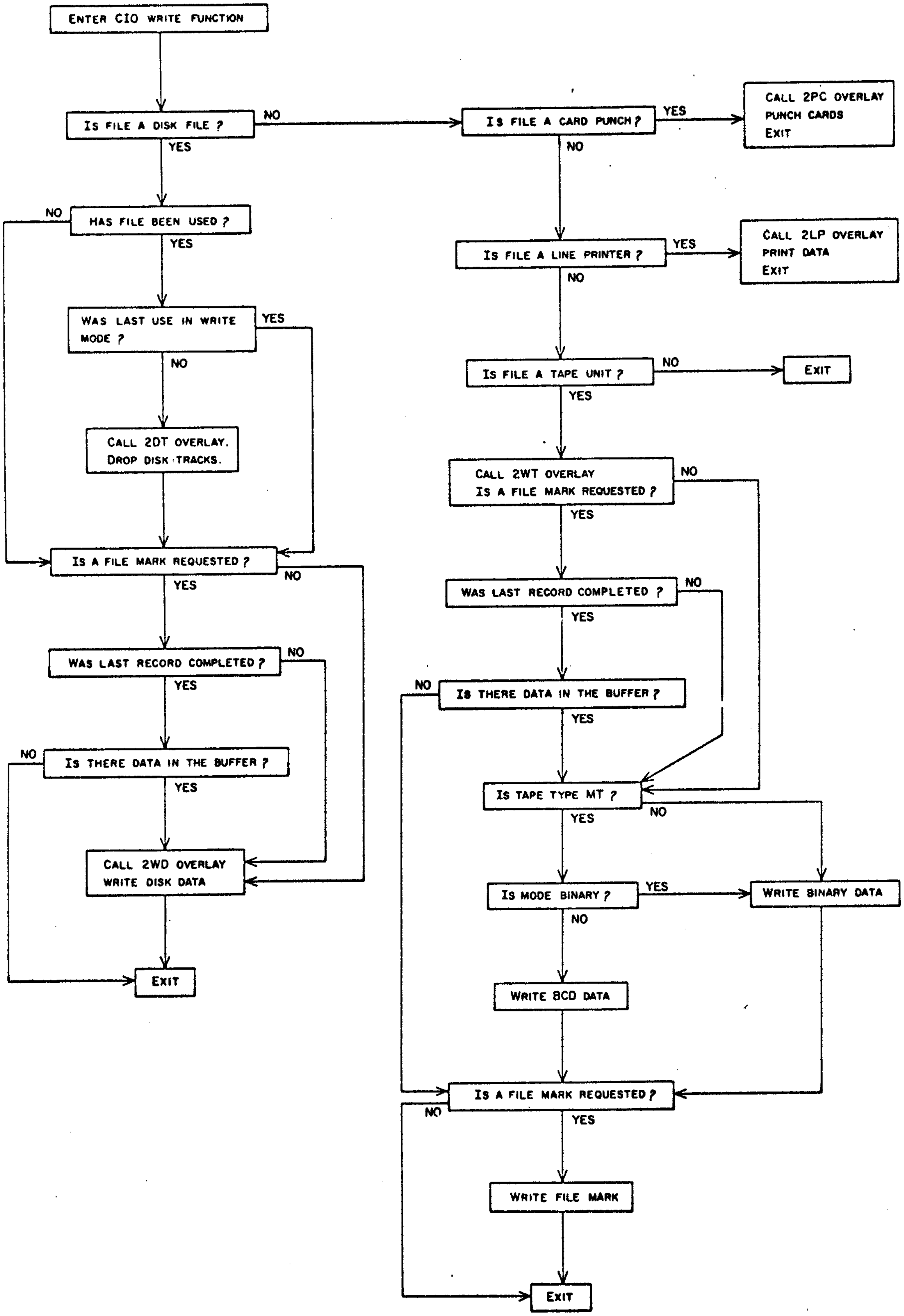
	first digit		second digit
0X	not used	X0	request coded read
1X	buffer I/O	X1	completed coded read
2X	end record	X2	request binary read
3X	file mark	X3	completed binary read
4X	backspace	X4	request coded write
5X	rewind	X5	completed coded write
6X	rewind/unload	X6	request binary write
7X	not used	X7	completed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

EXAMPLE:

READ	request to CIO	10	
	answer from CIO	11 21 31	full buffer end of record encountered end-of-file encountered
WRITE	request to CIO	14 24 34	dump as many complete records as possible empty buffer and write end of record empty buffer and write end of file
	answer from CIO	X5	where X from the call is unaltered


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
NO
CIO READ FUNCTION
IS A WRITE FUNCTION REQUESTED ?
YES
NO
CIO WRITE FUNCTION
IS A BACKSPACE FUNCTION REQUESTED ?
YES
NO
IS FILE A DISK FILE ?
YES
NO
CALL 2BD OVERLAY
BACKSPACE DISK
NO
IS FILE A DISK FILE ?
YES
IS FILE A TAPE UNIT ?
YES
NO
CALL 2BT OVERLAY
BACKSPACE TAPE
RESET FILE STATUS FOR REWIND
IS FILE A TAPE UNIT ?
NO
YES
SET READ MODE
SET IN = OUT = FIRST
SET READ MODE
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
NO
CALL 2RC OVERLAY
READ CARDS
EXIT
HAS FILE BEEN USED ?
NO
YES
IS FILE A TAPE UNIT ?
NO
YES
SET FILE MARK
EXIT
CALL 2RD OVERLAY
READ DISK DATA
CALL 2RT OVERLAY
IS MODE BINARY ?
NO
YES
IS TAPE TYPE MT ?
NO
YES
EXIT
READ BINARY TAPE
SET FILE MARK
EXIT
READ BCD TAPE
EXIT


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

CONTROL DATA CORPORATION

Development Division - Applications

DAYFILE

Chippewa Operating System

10/20/65

DAYFILE

INTRODUCTION

The dayfile is a combination accounting medium and job status record. It appears as a major display for the system console and is part of every job's output. Any message a programmer wishes to convey to an operator is passed through the dayfile. All control cards, error diagnostice, running times, and equipment assignments appear as a console display and are later sorted for a particular job's output.

A message may enter the dayfile from a central memory program or a peripheral routine. In the case of a central memory program, a peripheral package (MSG) is called to transfer the message from central memory to the PP message buffer and then to inform monitor that dayfile action is required. A peripheral routine need only put the message in the message buffer and let monitor take the appropriate steps. When monitor does sense that a message is ready, it transfers the message to the associated control point's dayfile area and then sends the message to the dayfile buffer in the proper format. This new entry is then picked up by the display program (DSD) and shown on the console.

STRUCTURE

The dayfile buffer status (DFB) is contained in word three of central memory. It points to a 1000_8 word buffer for dayfile entries and maintains the FIRST, IN, OUT, and LIMIT addresses. Each entry made in the dayfile consists of three parts.

1) The time that a message is sent.
2) The name of the job to which the message belongs.
3) The message of not more than six words.

All three parts are in separate words. Therefore, every dayfile entry is at least three words long but not more than nine. The time is read from word thirty of central resident and is in the form XX.YY.ZZ. where XX is hours, YY is minutes, and ZZ is seconds. At dead start this word is zeroed so it will reflect the time since dead start unless a "TIME" entry is made to DSD via the keyboard.

Monitor changes the spaces in the job name to blanks and terminates the field with a period. A zero byte ends the message so that word after word is transferred until the zero byte is found.

UPDATING BY CENTRAL PROGRAMS

In order for a central program to make entries into the dayfile, a peripheral program (MSG) is called to retrieve the message from central memory and inform monitor of the request. The location of the message and MSG (in left-justified display code) is inserted in RA+1 of the program. This causes MSG to be assigned to a PP and the message transferred to the PP's message buffer.

A check is made to insure that every character is a legal display code. If an illegal character is found, MESSAGE FORMAT ERROR is issued to the dayfile and the job is abandoned. Every entry made into the dayfile by a particular job advances a message count by one. In MSG this total is examined for an excess of 100_8 messages. No more than 6 words may be passed to the dayfile in one message. If either of these rules is violated, MESSAGE LIMIT is sent to the dayfile.

After the message is residing in the PP's message buffer, MTR is informed so that the message can be passed to the dayfile buffer. MSG is used by the Fortran compiler to enter the name of the program currently being compiled or executed.

UPDATING BY PERIPHERAL PROGRAMS

In a peripheral processor's resident program is a section of coding which copies a message from a transient program into the PP's message buffer. Each of these messages is assumed to have legal display codes and ended by a zero byte. The location of the message is in the A register upon entry to the routine. A return jump to location 530 will cause the message to be transferred to the PP's message buffer and then MTR is told of the request.

All transient programs use this method of making entries into the dayfile and each request will advance the message count at a job's control point, even though MSG is the only program which checks for a excess of the limit.

MTR - ISSUE DAYFILE

Whenever a PP has a message for the dayfile, the message is put into the message buffer and 0001 is inserted into the first byte of output register. MTR senses a request and begins dayfile updating procedures.

The message is passed from the PP buffer to an eight word area in the control point area. Word 30 of central memory which contains the current time is read into one word and the name of the job is put into another word. Next the message is copied until a zero byte is encountered and then all three sections are sent to the dayfile buffer. The PP output register is cleared to inform the PP that the message has been transferred. Only at this pointe is the message count increased by one so that every message is totalled.

IN and OUT are checked to see if 100_8 words (a full sector) of information is contained in the day file buffer. If there is, phase one dump flag is set. No additions may be made to the buffer when a dump flag is set.

MTR - COMPLETE DAYFILE

This function is issued by 1DJ (print package) or 1TD (tape dump package) when a job's output is being formatted. Its purpose is to remove all dayfile information from the buffer to the disk so that only the disk need be read when a job's dayfile is to be printed. The complete dayfile flag and 'dump phase one' flag are set. If the 'complete dayfile' flag is found to be set, then this is the second time through so it is cleared along with the output register.

Only the two MTR functions issue dayfile and complete dayfile, may set phase one dump flag.

MTR - CLOSE OUT

The dayfile buffer is dumped into the disk whenever a full sector of data is built up or whenever a job is to be printed. This process involves several steps, each of which set a flag for the subsequent phase. No entry may be made to the buffer when a dump flag is set.

On MTR's main loop a check is made to see if a dump flag is set. This flag is an address of the next phase and each phase is entered by a return jump. Every disk positioning request constitutes a different phase so that time is not wasted waiting on the disk.

Phase one requests channel 0 for the disk and phase two dump flag is set. MTR regains control and will continue its processing until the dump flag is checked again. This time phase two is entered via a return jump. If channel 0 is ready for use, a request for disk positioning to the proper track is issued and phase three dump flag is set. The current track and sector to be used by the dayfile is maintained by absolute coding. The 'update control byte' routines set the value of the current track and sector into the different dump phase locations directly.

Phase three checks channel 0 disk file status. The next sector must correspond to that set by 'update control byte' or an exit is made. One sector is written on the disk and the buffer parameters are updated accordingly. Then phase two flag is set. The buffer is dumped one sector at a time until a short sector is encountered. It is written on the disk but neither the buffer parameters nor the sector number are advanced. This scheme is used in order to maintain the dayfile as one record but still have all the information on the disk. Channel 0 is released via the output register and phase six flag is set if a spare track is assigned. If no spare track has been assigned, channel 0 is still released but phase four dump flag is set.

Phase six makes sure that the channel is released and clears the dump flag. This terminates dumping the dayfile buffer onto the disk so that normal processing may continue.

Phase four requests a track of MTR and sets phase five dump flag. When phase five is entered via a return jump, the spare track number is retrieved from the first byte of the message buffer and then the dump flag is cleared. This also completes the dayfile dump.

JOB DAYFILE LISTING

At the end of each job's output a complete history of each run during one dead start period is printed. 1DJ (print package) or 1TD (tape dump package) requests MTR to dump the dayfile buffer contents on the disk in the manner just previously described. Next, one sector of the dayfile is read and it is searched for the job's entries by 2SD (search dayfile).

Since the time a message is issued appears in the word before the job name, every word of the sector is checked for the proper job name. If the word does not match, it is copied into the peripheral buffer but its parameters are not advanced. When the name finally matches, the time has already been copied into the peripheral buffer so the job name is added in the next word. Then the subsequent message is transferred until the zero byte is encountered.

Control fluctuates between the dump package, i.e. 1DJ or 1TD, which reads a sector of the dayfile, and 2SD, which searches it for a particular job name. The dayfile is searched in this manner until a short sector is found. When it is encountered, a MTR function requesting assignment of PP time to the control point is made. This computes the total PP running time and stores it in word 24 of the control point area. 2SD converts this time to decimal seconds and then sends out a dayfile message "PP XXXX SEC".

A top of form request is made as the first entry into a circular buffer in central memory. The peripheral buffer containing the dayfile information for this job is copied to the circular buffer. An entry of the same type, "PP XXXX SEC", that was sent to the dayfile is added to the circular buffer. Now the job's dayfile is complete and ready for printing.

NOTES

1. The dayfile is the first entry in FNT. It is set from the library tape and is of common type so that any program may access it.

2. Any message sent to the dayfile also appears as a console message (line 3 of the control point display)

CONTROL DATA CORPORATION

Development Division - Applications

DISK ROUTINES AND OVERLAYS

Chippewa Operating System

11/1/65

Disk Routines and Overlays

Contents

DISK ROUTINES AND OVERLAYS

Introduction

In the Chippewa Operating System, there is no single system element used to perform disk operations for all other elements of the system. Instead, each system element performs its own disk operations. This, while requiring additional coding for each of the system elements using the disk, eliminates the need for a request queueing and priority scheme required by the use of a single system element to process all disk operations. In addition, the housekeeping required by a disk subroutine in one system element can overlap, to some extent, a disk operation being performed by another system element. Among the system elements which perform disk operations are:

- peripheral processor resident (reads transient programs from the disk library)
- MTR (writes the contents of the dayfile buffer to the disk)
- some transient programs (read overlays from the disk)

Disk operations for external users are performed via the overlays 2WD (write disk), 2RD (read disk), and 2BD (backspace disk). These overlays are called by CIO when a disk operation is requested by a central processor program. In addition, these overlays are used by certain transient programs to perform disk operations. Thus, 1LJ and 1LT call 2WD when loading jobs from the card reader and a tape unit, respectively, while 1DJ and 1TD call 2RD when transferring job output to the printer or a tape unit.

Regardless of where in the system they are performed, disk operations are similar: this discussion will therefore be limited to the overlays 2WD, 2RD, and 2BD. Before discussing these routines a short review of the physical characteristics of the 6603 disk file is in order.

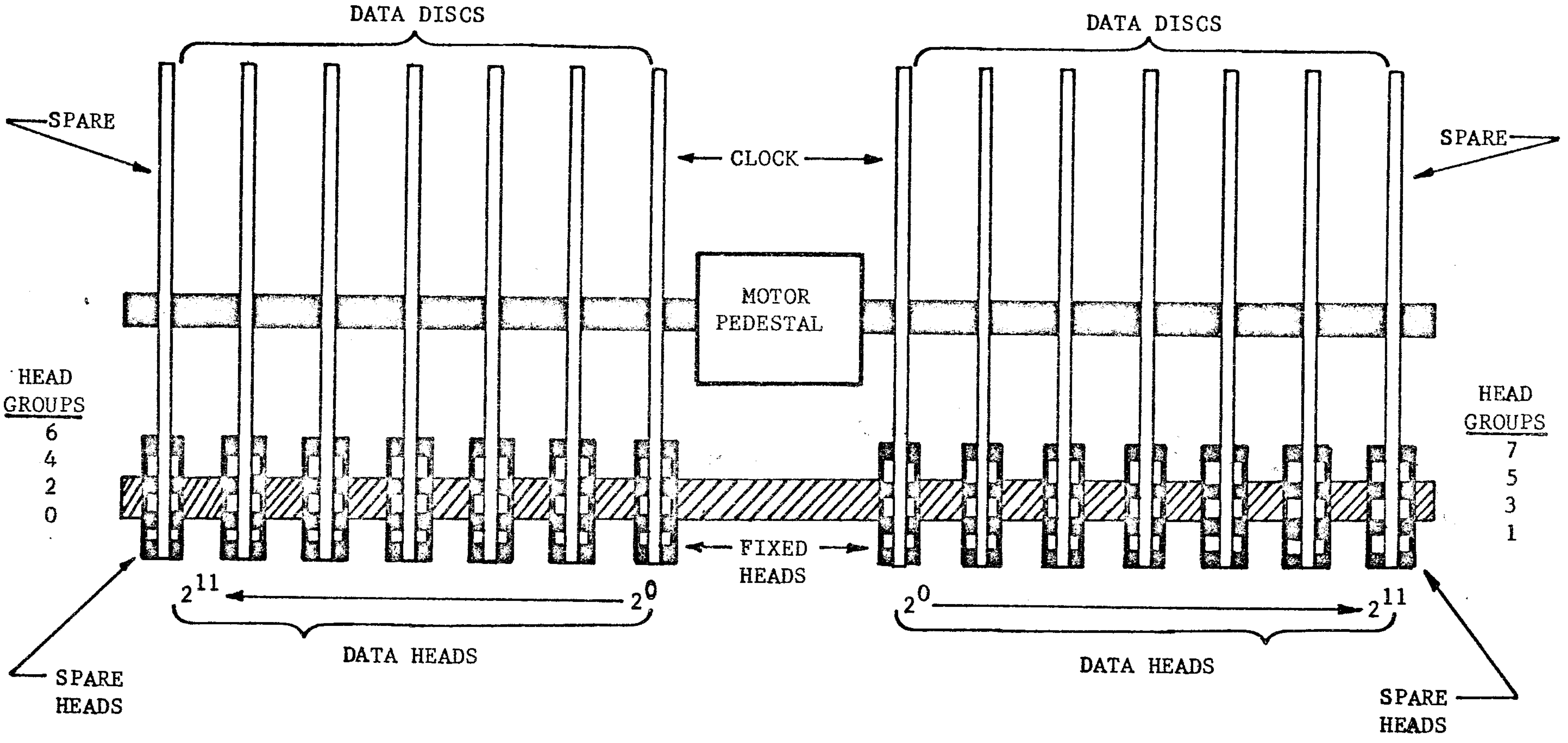
6603 Disk File: Description and Organization

The 6603 Disk File contains fourteen disks, each coated on both sides with magnetic oxide. Thus, there are a total of twenty-eight recording surfaces. On two of these surfaces timing tracks are recorded, two are used for spares, and twenty-four are used for recording data (see figure 1). All fourteen disks are mounted (in a vertical plane) on a common axis and rotate at a speed of approximately 900 revolutions per minute. Twelve of the data surfaces are on the right side of the unit, and twelve are on the left. Information is recorded on the disk in 12-bit bytes: each bit in a 12-bit byte is recorded on a separate disk surface.

Associated with each disk surface is a set of four read/write heads (see figure 2). An assembly consisting of a rocker arm and a head bar fits between each pair of facing disk surfaces. The head bar holds two sets of four heads, one set for each of the two facing surfaces. The read/write heads are mounted on this head bar in a fixed position relative to each other. The rocker arm-head bar assemblies for all disks mount on a common bracket which can be rotated. This rotation moves all the head bars simultaneously (with the exception of the heads accessing the timing track surfaces: these heads are fixed).

The disk surface is divided into four zones. A zone is that portion of the disk surface transversed by one of the four heads associated with that surface as the head (on its head bar-rocker arm assembly) moves through its maximum angular rotation. A byte may be written on the twelve data surfaces on the right side of the disk file or on the twelve data surfaces on the left side of the disk file: on either side, a byte may be written in any one of four zones. On each side of the disk file and for each zone on side, a single set of twelve read/write heads are used to record a byte (see figure 1). This set of twelve heads is called a head group. There are four head groups for each of the two sets of twelve disk surfaces: a total of eight head groups.

Each zone contains 128 tracks. A track is the recording path available to a given head group in a given position as the disk makes a complete revolution. To move from one track to another requires a physical



ALL HEADS (EXCEPT FIXED HEADS) MOVE TOGETHER

6603 DISC FILE

Figure 1

movement, or repositioning, of the head bar-rocker arm assemblies. At a given position, each head group accesses the same track in its zone. Thus, if head group 2 is positioned to track 125, the other 7 head groups are also positioned to track 125.

Tracks are divided into sectors: a sector is the smallest addressable segment of a track. There are 128 sectors in each of the tracks in the two outer zones. In the two innermost zones, there are only 100 sectors per track because of the reduced track length near the center of the disk compared to the track length available near the outside edge. A sector contains 351 bytes (each bit in a byte is recorded in one of 12 corresponding sectors across 12 disk surfaces). The first four bytes recorded are reserved for use by the controller: They provide a time lag between consecutive sectors and contain all zero bits. After the last data byte has been written, the controller writes a longitudinal parity byte. The sector format is illustrated in figure 3. Of the 351 bytes in a sector, then, five are used by the controller: The remaining 346 bytes may be used for data. Normally, 320 bytes (the equivalent of 64 central memory words) are used for data.

The number of words read from or written to the disk is solely a function of the word count specified in the IAM or OAM instruction. It is possible to read or write more than one sector at a time; it is possible to read or write in the group switch gap; it is possible for a read or write to wrap around on the same track. A read or write operation always begins at the beginning of a sector. When a write is initiated, the disk controller inserts four zero bytes before the data and inserts a parity byte after the last data byte. (The parity byte is not necessarily in the last byte position in a sector.) When a read is initiated, the controller assumes that the first four bytes are zero bytes, and does not pass these on to the data channel. When the word count in a read has been reduced to zero, the controller assumes that the next byte to be read is the parity byte. Thus, any attempt to read a number of bytes different than the number of bytes written will invariably create problems due to the interpretation of zero bytes and parity bytes as data and vice versa. For this reason, regardless of the amount of data to be recorded, a fixed number of bytes is written in each sector,

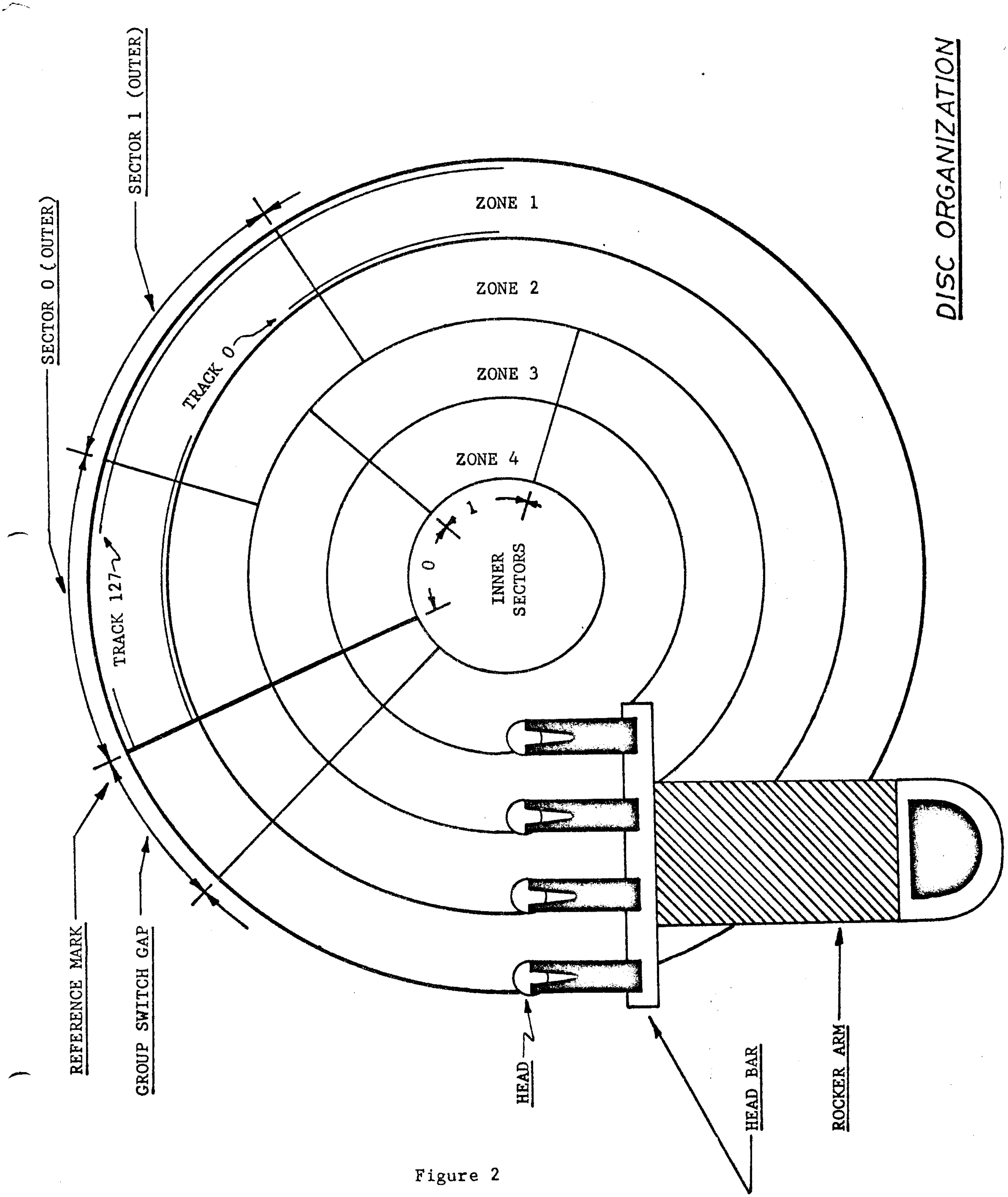
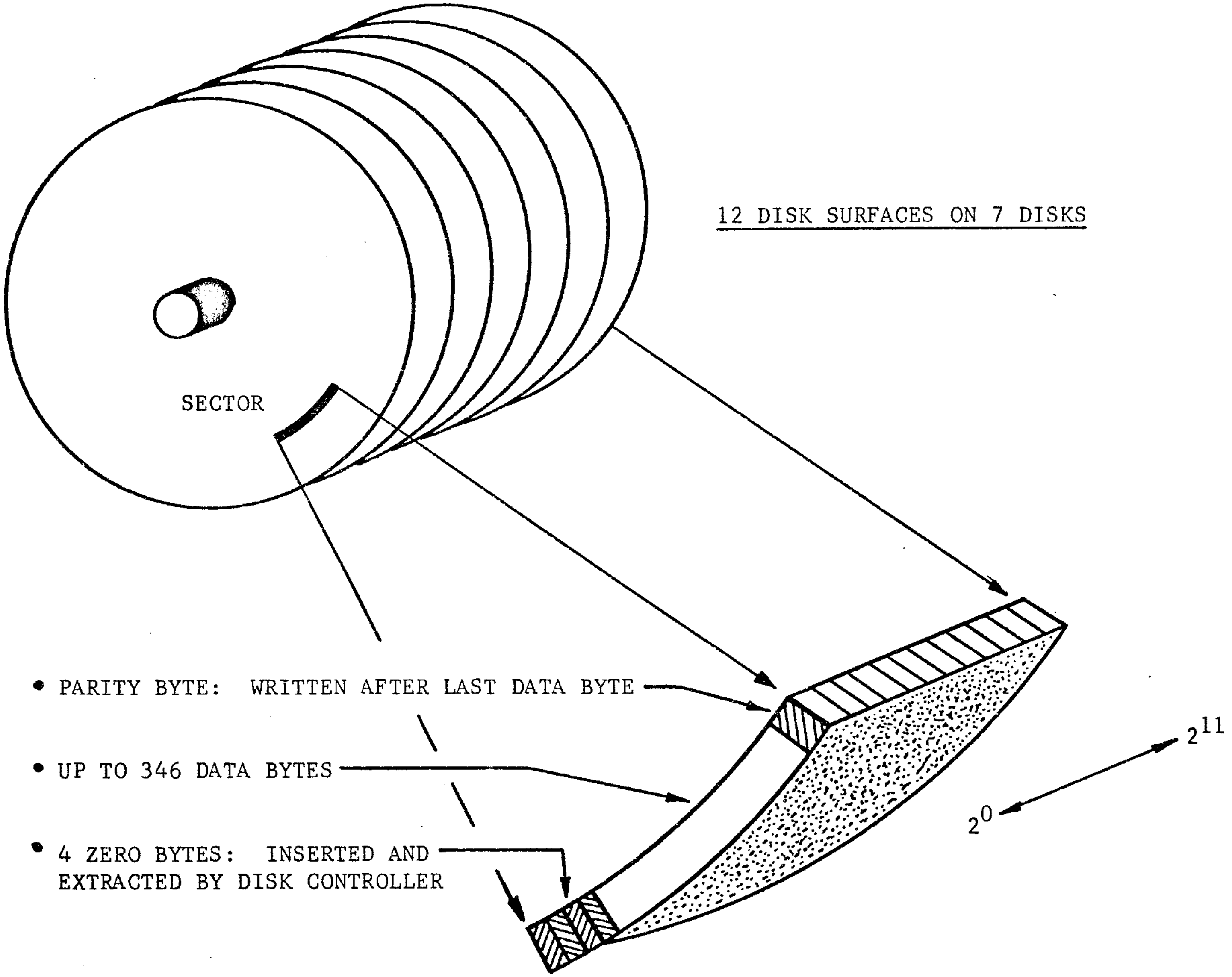

DISC ORGANIZATION

Figure 2



12 DISK SURFACES ON 7 DISKS
SECTOR
• PARITY BYTE: WRITTEN AFTER LAST DATA BYTE
• UP TO 346 DATA BYTES
• 4 ZERO BYTES: INSERTED AND EXTRACTED BY DISK CONTROLLER
2^{11}
2^0

SECTOR FORMAT: 6603 DISK FILE

Figure 3

and only one sector is written at a time (i.e., data is recorded in physical records of one sector).

A reference mark on the disks containing the timing tracks defines the beginning of sector 0 in all four zones. Beyond this point, the starting point of sectors in the two inner zones does not coincide with the starting point of sectors in the two outer zones (see figure 2). The clock surfaces contain timing tracks for each zone. As the disk rotates, one of these timing tracks (depending on which head group is selected) drives a cell counter. This counter in turn triggers a sector counter. Both counters are initialized when the reference mark is detected. The cell counter is incremented as the timing track is read: When it reaches a count of 351, it is reset and the sector count advanced. The controller compares the sector number specified in a read or write function code: When equality is obtained, the read or write operation is initiated. The contents of the sector counter appear in the low-order 7 bits of the status response.

6603 Disk File: Timing Considerations

The rotational speed of the disk is approximately 900 revolutions per minute, corresponding to a revolution time of about 66 milliseconds. The time required to read or write a byte is approximately 1.4 microseconds on the two outer zones and 1.8 microseconds on the two inner zones. In the outer zones, then, a sector passes under the heads every 490 microseconds. It requires a minimum of 325 microseconds to transfer the 64 central memory words in a sector from peripheral processor memory to central memory, and, because of memory and pyramid conflicts, will probably require longer. A single peripheral processor cannot maintain a continuous data flow between consecutive sectors on the disk and central memory.

If the programmer wishes to read or write in a given sector, he simply issues the appropriate function code and, when the sector comes under the heads, the operation is initiated. The programmer may prefer to minimize the time spent waiting for this sector by sensing (via a status request) the position of the disk. Timing considerations make it impossible to sense for a given sector and then initiate an operation in that sector: If one wishes to read or write sector N, then sector N-2 should be sensed in order to assure that a revolution will not be lost.

There are two types of delays which are of concern to the disk programmer. One of these is the positioning delay: The time required to move the heads to a new track. When a track select function has been received by the disk controller and positioning initiated, a delay determined by counting 4 reference marks is provided to permit the head assembly to stabilize. Thus, depending on when positioning is initiated, up to 264 milliseconds may be required. During positioning, a status request will receive a "NOT READY" reply.

The second type of delay is the switching delay encountered when a different head group is selected. When head group switching is initiated, the controller provides a one millisecond delay to allow the circuits to stabilize: Furthermore, reading or writing cannot be initiated until a reference mark is detected. Thus, depending on when the head group select function is issued, up to 66 milliseconds may be required for head group selection.

Between the last sector in a track (sector 127 in the outer zones, sector 99 in the inner zones) and the first sector (sector 0) on that track is an area called the group switch gap (see figure 2). This area is approximately equivalent to three sectors in size. It is provided to accommodate the minimum 1 millisecond switching delay. A programmer can thus read or write the last sector in a track, select a new head group, and read or write sector zero of the new track without incurring a delay.

The function code for head group selection is 160X, where X is the head group number (0-7). It is possible to vary the second octal digit in this function code (normally zero) from 1 to 7: In doing so, the manner in which the data signals from the disk are sampled is varied. Use of the feature is reserved for error routines.

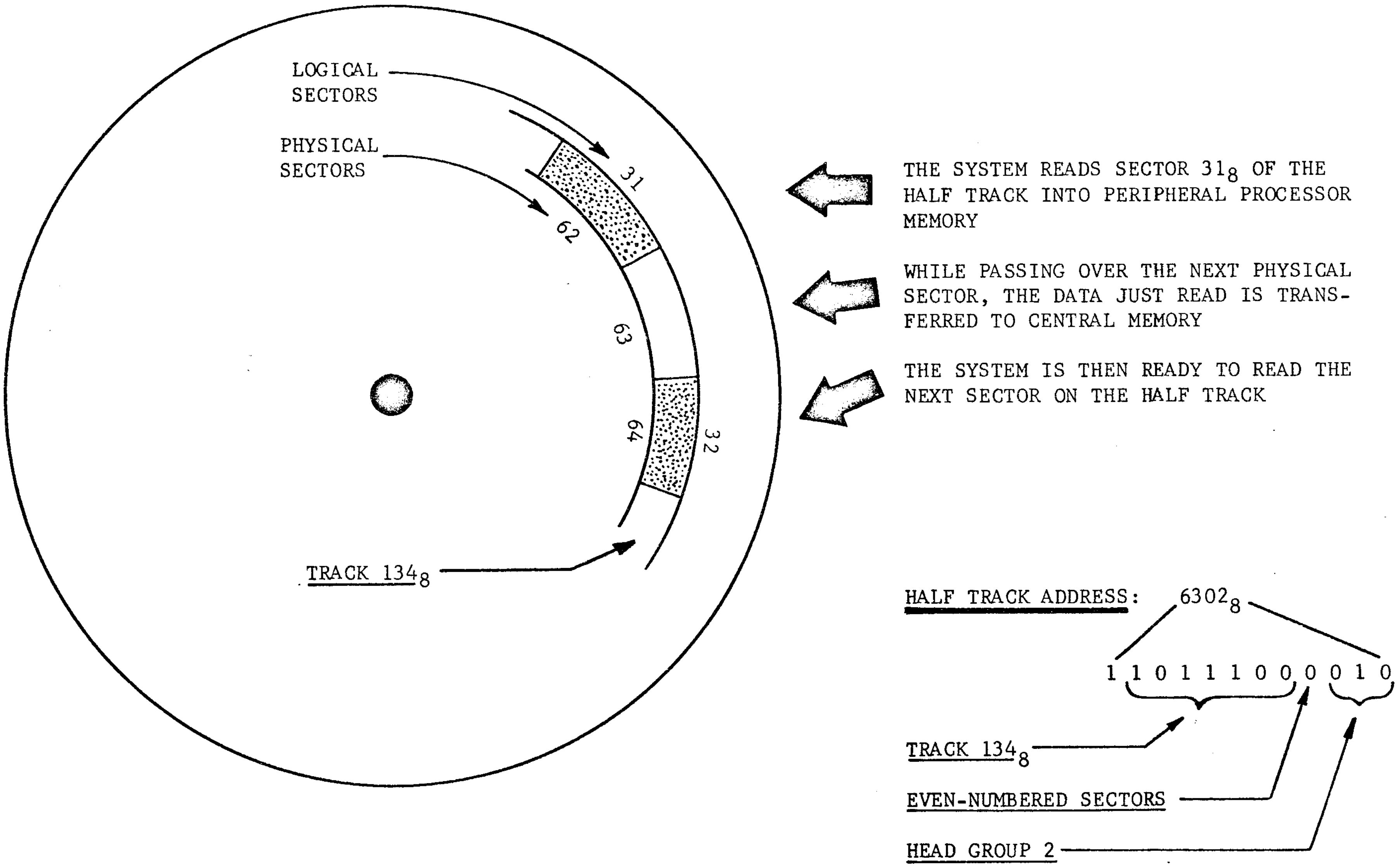
6603 Disk File: Data Capacity

There are 128 physical positions of the heads: At any one position, a track may be accessed by selecting one of eight head groups. Thus, the disk has a total of 8 x 128 = 1024 tracks. Of the eight head groups, four cover inner zones and four cover outer zones. In the inner zones, there are 100 sectors per track: In the outer zones, there are 128 sectors per track. Therefore, 512 tracks each contain 100 sectors while the other 512 tracks each contain 128 sectors. The disk file thus contains 116, 736 sectors. In normal use, up to 64 central memory words are recorded in a sector. The capacity of the 6603 disk file is thus approximately 7.5 million central memory words.

Chippewa Operating System Disk Usage

As we have seen, a single peripheral processor cannot maintain a continuous data flow from consecutive disk sectors to central memory. Therefore, the Chippewa Operating System uses a half track scheme in its disk operations. A half track is composed of either the odd-numbered or the even-numbered sectors in a track. In a disk operation, the system reads or writes alternate sectors, transferring data to or from central memory while passing over the intervening sector. Since the disk contains 1024 physical tracks, the equivalent half track capacity is 2048. The allocation of half tracks is controlled by MTR: disk write routines obtain half track addresses from MTR via the Request Track function. MTR maintains a table called the Track Reservation Table (TRT) which contains an entry for each half track on a disk. On receipt of the Request Track function, MTR searches the table for an unassigned half track, and returns the half track address to the requestor in the upper byte of the Message Buffer. If no half track is available, a zero address is returned to the requestor. A half track is never split between files: thus, the half track is the smallest unit of storage allocated on the disk.

The format of the half track address, and its relationship to physical disk addresses, is illustrated below.



THE SYSTEM READS SECTOR 31_8 OF THE HALF TRACK INTO PERIPHERAL PROCESSOR MEMORY

WHILE PASSING OVER THE NEXT PHYSICAL SECTOR, THE DATA JUST READ IS TRANSFERRED TO CENTRAL MEMORY

THE SYSTEM IS THEN READY TO READ THE NEXT SECTOR ON THE HALF TRACK

HALF TRACK USE: AN EXAMPLE

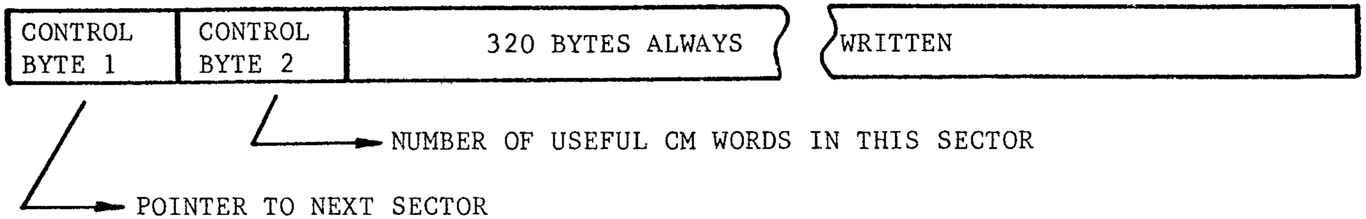
Figure 4

```
1XXXXXXXXXXX
  |    | |___ head group number (0-7₈)
  |    |_____ "1" if odd sectors, "0" if even sectors
  |__________ track number (0-177₈)
```

Sector numbers maintained by the system (such as the Current Sector in an FST entry) are logical sector numbers, and refer to a sector within a half track. In the outer zones, sectors within a half track are numbered 0-77_8: In the inner zones, sectors within a half track are numbered 0-61_8. To convert a logical sector number to a physical sector number, the system shifts the logical sector number left one place and inserts the 2^4 bit from the half track address into the low-order bit position. For example, consider logical sector 77_8 (63_{10}) in a half track composed of the odd-numbered sectors in a physical track. In this case, the 2^4 bit of the half track address will be a "1". By shifting the logical sector left one place and inserting the "1" bit from the 2^4 bit position of the half track address, we obtain 177_8 (127_{10}) for the physical sector number. For the remainder of our discussion, a reference to "sector number" will refer to the logical sector number unless otherwise described.

For files recorded on the disk, the physical record is, of course, the sector. A logical record may be composed of several sectors. The format of the physical record is shown in figure 5. 502_8 bytes are always written in each sector. The first two bytes written are control bytes: the remaining 500_8 bytes are data bytes. Control byte 2 contains the number of useful central memory words in this sector: If control byte 2 contains 100_8, all 500_8 bytes in this sector contain useful information. A sector in which control byte 2 contains less than 100_8 is called a short sector, and is interpreted as a record mark. A logical record may comprise several full sectors, but is always terminated by a short sector. If the data to be recorded as a logical record is a multiple of 100_8 CM words, the system will write, as the record mark, a sector in which control byte 2 contains zero.

Control byte one points to the next physical record in this file. If the next sector is on the same half track, then this byte contains the



NUMBER OF USEFUL CM WORDS IN THIS SECTOR

POINTER TO NEXT SECTOR

- SECTOR NUMBER (0 - 77_8) IF ON SAME HALF TRACK
- HALF TRACK NUMBER IF ON ANOTHER HALF TRACK

CONTROL BYTE 1	CONTROL BYTE 2	RECORD
NON-ZERO	100_8	"FULL" SECTOR: PART OF A LOGICAL RECORD
NON-ZERO	NON-ZERO, $<100_8$	"SHORT" SECTOR: PART OF A LOGICAL RECORD; RECORD MARK
NON-ZERO	ZERO	"SHORT" SECTOR: RECORD MARK
ZERO	ZERO	FILE MARK

DISK FILE PHYSICAL RECORD FORMAT

Figure 5

number of that sector. If the next sector is on another half track, then this byte contains the half track address for that half track. (The file would be continued beginning with sector zero of the new half track.)

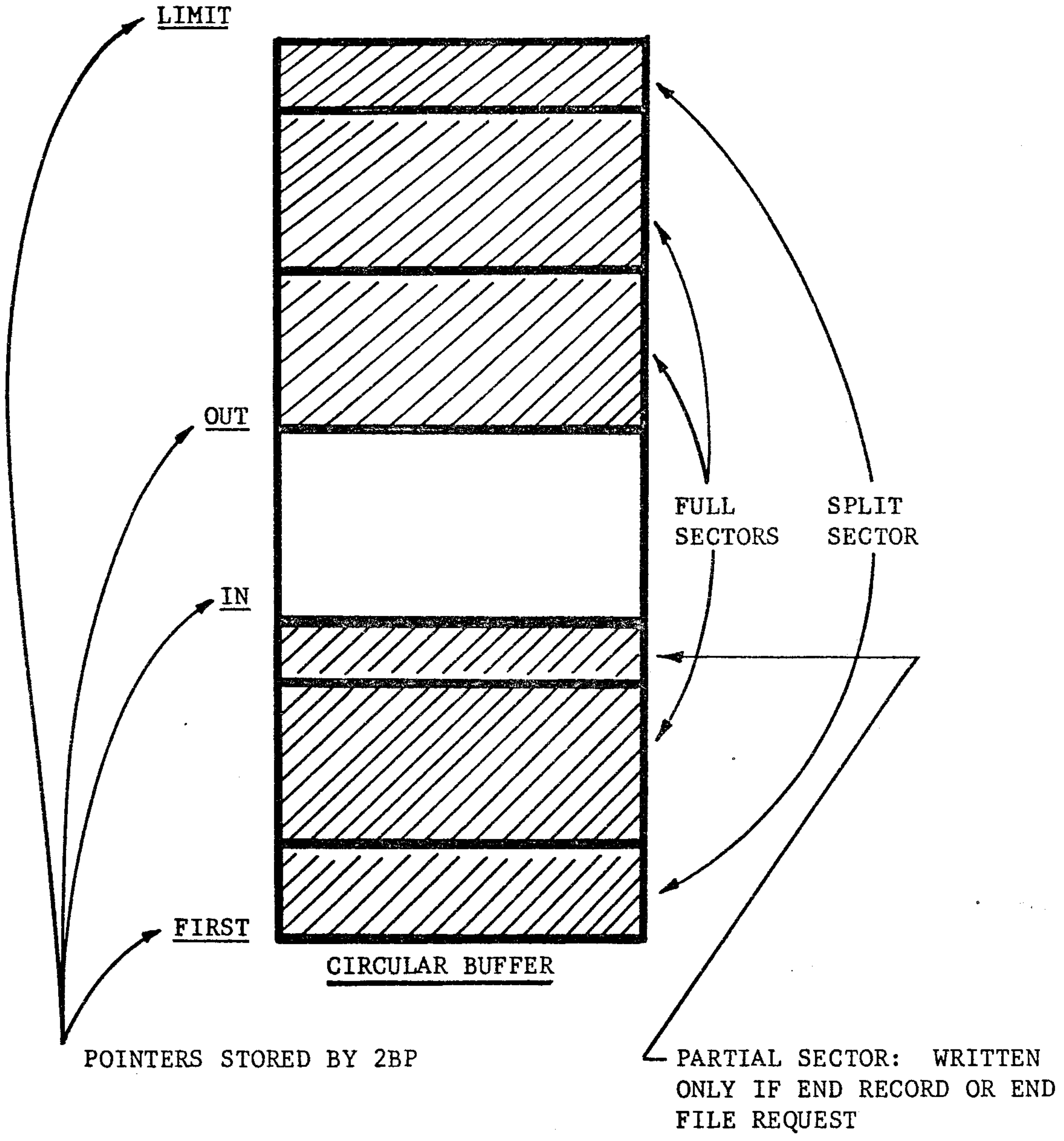
At the end of each write operation, the system writes a file mark. The Current Sector byte of the FST entry is not incremented to reflect this file mark sector, so the effect is equivalent to writing a file mark and backspacing over it. On the disk, a file mark is a sector in which both control bytes contain zero.

The Disk Write Overlay, 2WD

Disk write requests by users are executed by CIO's overlay 2WD. This overlay is also used by 1LJ and 1LT in loading jobs on the disk. Before calling 2WD, CIO calls the 2BP overlay to check the legality of the buffer parameters FIRST, IN, OUT, and LIMIT. After checking these parameters, 2BP searches the File Name Table for the file name specified in the CIO call (i.e., in the first word of the argument list). When found, 2BP stores the address of the corresponding FST entry. Should the file name not be found in the FNT, 2BP constructs an FNT entry for this file. Finally, 2BP clears the 2^0 bit in the buffer status byte of the FST entry to reserve the file.

CIO then calls 2WD. (Refer to the flow chart on page A-1.) 2WD reads the FST entry for the file and extracts the equipment number from byte one. The equipment number is added to the EST base address, and the EST entry read. The channel number from byte 2 of the EST entry is then inserted in the appropriate I/O instructions.

The output data in the circular buffer may appear as a contiguous block, or may wrap around the buffer, as illustrated in figure 6. In computing the total number of sectors in the circular buffer, then, the 2WD routine first subtracts OUT from IN. If the difference is positive, then this difference is the total number of words to be written, and 2WD shifts off the lower six bits of this word count in order to obtain the equivalent number of sectors. If OUT-IN is negative, the value of LIMIT is added to the difference and FIRST subtracted to obtain the



BUFFER PARAMETER PROCESSING

1. COMPUTE TOTAL NUMBER OF WORDS IN OUTPUT AREA

2. COMPUTE TOTAL NUMBER OF SECTORS IN OUTPUT AREA BY SHIFTING TOTAL WORD COUNT RIGHT 6 PLACES

3. COMPUTE NUMBER OF WORDS BETWEEN OUT AND LIMIT

4. COMPUTE NUMBER OF SECTORS BETWEEN OUT AND LIMIT BY SHIFTING OUT-LIMIT WORD COUNT RIGHT 6 PLACES

5. EXTRACT LOW-ORDER 6 BITS OF OUT-LIMIT WORD COUNT: THIS GIVES THE NUMBER OF WORDS IN THAT PART OF THE SPLIT SECTOR BETWEEN OUT AND LIMIT

6. SUBTRACT NUMBER OF WORDS COMPUTED IN (5) FROM 100_8 TO GET NUMBER OF WORDS TO BE READ FROM THE BUFFER BEGINNING AT FIRST IN ORDER TO COMPLETE THE SPLIT SECTOR

7. SET UP INSTRUCTIONS FOR PROCESSING THE SPLIT SECTOR

CIRCULAR BUFFER PARAMETER PROCESSING - 2WD OVERLAY

Figure 6

total word count and, from that, the equivalent number of sectors.

Regardless of whether the data is contiguous or wraps around the buffer, 2WD proceeds on the assumption that the data does wrap around, and proceeds to compute the values needed to process the wraparound case. The steps involved are listed in figure 6. These values, although always computed, are not required in the contiguous case: in either case, the terminal path is entered when the total sector count is reduced to zero. By computing these values regardless of whether the data is contiguous in the buffer or wraps around the buffer, computations during the period when the disk is actively in use are reduced.

Next, 2WD picks up the channel number from the EST entry and requests reservation of that channel from MTR. The Current Track byte of the FST entry for this file is then examined. If this byte is zero, then this file has not previously been used. A half track assignment is requested from MTR: MTR returns a half track address to the requestor in byte one of the first word in the message buffer. If no half track is available, MTR will return a zero byte to the requestor: 2WD then inserts an error message in the dayfile and aborts the control point after dropping the channel reservation. 2WD now has the address of the half track where the next operation is to be performed, and proceeds to position the disk to this half track. This half track address is compared with byte 2 of the TRT pointer word for this disk, and repositioning or head group selection performed only if required. Byte 2 of the TRT pointer is then updated.

2WD next requests another half track assignment from MTR. This half track is a spare: by keeping it available, it is possible for 2WD to switch head groups within the group switch gap if this action should be required when the end of the current half track is reached.

The transfer of data from the buffer to the disk then begins. 2WD reads 100_8 words from central memory into peripheral processor memory, sets control bytes one and two, and then writes the completed sector to the disk. As each sector is written, the number of the sector is examined to determine if the end of the half track is reached. To do this, 2WD compares the sector number with byte 4 of the TRT pointer word (if head group number = 0-3) or byte 5 of the TRT pointer word (if head group number = 4-7). These bytes contain the values 100_8 and 62_8, respectively.

If the end of the half track has been reached, 2WD positions the disk to the spare half track: again, the half track address is compared with byte 2 of the TRT pointer word and positioning or head group selection performed only if required. After initiating any repositioning which might be required, 2WD requests a spare half track from MTR.

2WD continues reading 100_8-word blocks from central memory and writing them to the disk until it recognizes that there is not enough data in the circular buffer for a complete sector. (Some part of a sector may still, however, remain.) 2WD then examines the buffer status contained in byte 5 of the FST entry to see if an end record was requested (2^4 bit = 1). If an end record was requested, 2WD writes a short sector to the disk. If any data remained in the circular buffer, it will be written in this short sector: otherwise, control byte 2 will simply be set to zero.

After the last data sector has been written to the disk, 2WD writes a file mark - a sector with both control bytes equal to zero. The Current Sector byte of the FST entry is not, however, incremented to reflect the writing of this file mark: the next write to this file will write over the file mark sector. After the file mark has been written, 2WD requests MTR to drop the spare half track assignment and to release the channel reservation.

If no end record function was requested, 2WD simply updates the OUT pointer before returning control to CIO: There may still be some data in the circular buffer. If an end record function was requested, no data remains in the buffer: 2WD therefore sets IN = OUT = FIRST to indicate that the buffer is empty.

When control is returned to CIO, CIO sets the 2^0 bit of the buffer status in the FST entry to 1 to indicate that the file is no longer in use, and sets the 2^0 bit of the buffer status in the calling program's argument list to 1 to indicate to the calling program that the operation has been completed.

The Disk Read Overlay, 2RD

Disk read requests by users are executed by CIO's overlay 2RD. This overlay is also used by 1DJ and 1TD. The processing performed by 2BP in this case is identical to that performed in the case of 2WD. On entry, 2RD reads the FST entry for the file, picks up the equipment number from byte one, and uses this number to obtain the EST entry. The channel number from the EST entry is then set in the I/O instructions.

2RD then proceeds to compute the number of sectors which can be loaded into the circular buffer. If there is not room for a full sector, control is returned to CIO. The data to be read may fit in the buffer in a contiguous block, or may wrap around the buffer. The computation of the values (total word count, total sector count, etc.) used in controlling the transfer of data to the buffer is performed in a manner similar to 2WD. Again, the wraparound case is assumed.

The Current Track byte of the FST entry is examined. If this byte is zero, the file has not been used before and so contains no data. 2RD sets the buffer status to indicate a file mark and returns control to CIO.

2RD requests a channel reservation from MTR and positions the disk to the half track address contained in the FST entry's Current Track byte. As in all disk routines, the half track address is compared with the disk position specified in the TRT pointer, and repositioning or head group switching performed only if necessary.

2RD then uses the Current Sector byte of the FST entry to construct the read function code, and reads the specified sector into peripheral processor memory. A status request is then issued, and the response is examined to determine if a parity error occurred. In the event of a parity error, the system rereads the sector three times; once using the normal sampling method and twice at varied sampling margins. If the parity error re-occurs in each of the rereads, 2RD inserts an error message in the dayfile and stops (via a UJN 0 instruction). Since the halt occurs without the disk channel being released, all system activity will shortly cease (if this disk is the system disk, disk 0). A dead start load will be necessary to reinitiate processing.

If the read was successful, 2RD examines the high-order six bits of control byte one: if these bits are zero, then this control byte contains a sector number, while if these bits are non-zero, this control byte contains a half track number. In the latter case, 2RD positions the disk to the new half track address. While any repositioning or head group switching which might be required is in process, 2RD transfers the number of words specified in control byte 2 from peripheral processor memory to the circular buffer, and updates the values used in controlling the transfer. If the sector just read was a full sector (100_8 CM words of data), and if there is enough room in the circular buffer for another full sector, 2RD loops to read the next sector from the disk.

If the last sector read was a short sector, then the end of a logical record has been reached, and the buffer status is set to reflect a record mark. If the end of logical record has been reached, or if there is not enough room in the circular buffer for a full sector, 2RD requests MTR to release the channel reservation, updates the IN pointer in the calling program's argument list, and returns control to CIO. CIO updates the buffer status in the FST entry to release the file reservation, and updates the buffer status in the calling program's argument list to indicate that the operation has been completed.

If, after reading the last logical record in a file, the calling program issues another read to the file, the file mark will be read. The processing proceeds as described above: 2RD reads a sector whose address is specified in the Current Track and Current Sector bytes of the FST entry. Since control byte 2 is zero, 2RD recognizes this as a short sector, sets the buffer status to reflect a record mark, and releases the channel. 2RD then examines control byte one; since this contains zero, the file mark is recognized and the buffer status set accordingly before returning control to CIO.

The Backspace Disk Overlay, 2BD

Disk backspacing may take the form of a BCD backspace or, more commonly, a binary backspace. In either case, it is desired to backspace over a logical record, and it is assumed that any backspacing over logical records in the buffer has been done by the calling program. Backspacing over the physical records which may constitute a logical record is essentially a matter of backspacing over two sectors and then reading a sector.

2BD uses a subroutine to backspace over a sector. (See flow chart on page A-5.) This subroutine examines the Current Sector byte of the FST entry, and, if non-zero, subtracts one from this number and exits. This is equivalent to backspacing over one physical record (i.e., one sector). If the Current Sector number is zero, then the preceding physical record is on another half track. In this case, the subroutine stores the Current Track byte from the EST entry for this file, since it will have to search the file for a sector which has this half track address contained in control byte one.

The subroutine rewinds the file by picking up the Beginning Track byte from the FST entry. (Should the Beginning Track byte be equal to the Current Track byte, the subroutine exits, since this indicates that the system has backspaced over all physical records in this file.) After rewinding the file, the subroutine reads each sector in the file until it finds a sector with the desired half track address in control byte one. The number of this sector is then stored, and control returned to the calling routine. A backspace operation on a file of any size may take considerable time if it should become necessary to rewind the file and search forward.

A binary backspace on the disk consists of backspacing over two sectors (using the subroutine described above) and reading a sector until a short record is found, indicating the end of a logical record. 2BD sets the circular buffer pointers IN and OUT equal to FIRST, and returns control to CIO. CIO updates the buffer status in the FST entry and in the calling program's argument list before exiting.

It is also possible to issue a BCD backspace to the disk. For the disk, as for 1" tape (but not for 1" tape), a logical BCD record consists of a series of central memory words presumably containing display code data, terminated by a central memory whose low-order byte (byte 5) is zero.

The BCD backspace begins with the computation of the amount of data left in the buffer as a result of the last read. This quantity, referred to as D, is equal in IN-OUT if the data in the buffer is contiguous, or IN-OUT + LIMIT-FIRST if the data wraps around the buffer. This data was left in the buffer as a result of the last read, and may have been stored on the disk in several sectors. The system assumes that the calling program will backspace within the buffer, and so, before beginning a logical BCD record backspace on the disk, 2BD will backspace the disk a number of sectors equivalent to the amount of data contained in the buffer. This quantity is represented by D.

2BD therefore backspaces over a sector (by the same subroutine used in binary backspacing and described earlier) and reads that sector into peripheral processor memory. The sector length in control byte 2 is then compared with D: if less than D, then this sector is assumed to contain data which has already been read into the buffer. 2BD then decreases D by this amount, backspaces over this sector and the sector preceding it, and then reads a sector. The process of backspacing, reading, and reducing D is repeated until a sector is read whose length is greater than the present value of D: this sector could not entirely be part of the read data in the buffer, and so must be searched for a logical record. 2BD transfers this sector from peripheral processor memory to the circular buffer beginning at FIRST. If D is still non-zero, then part of this sector contains data residing in the buffer at the time the backspace was requested, and presumably has been searched by the calling program: 2BD therefore sets the OUT pointer to FIRST + sector length - D. At the same time, the IN pointer is set to reflect the transfer of the sector to the buffer.

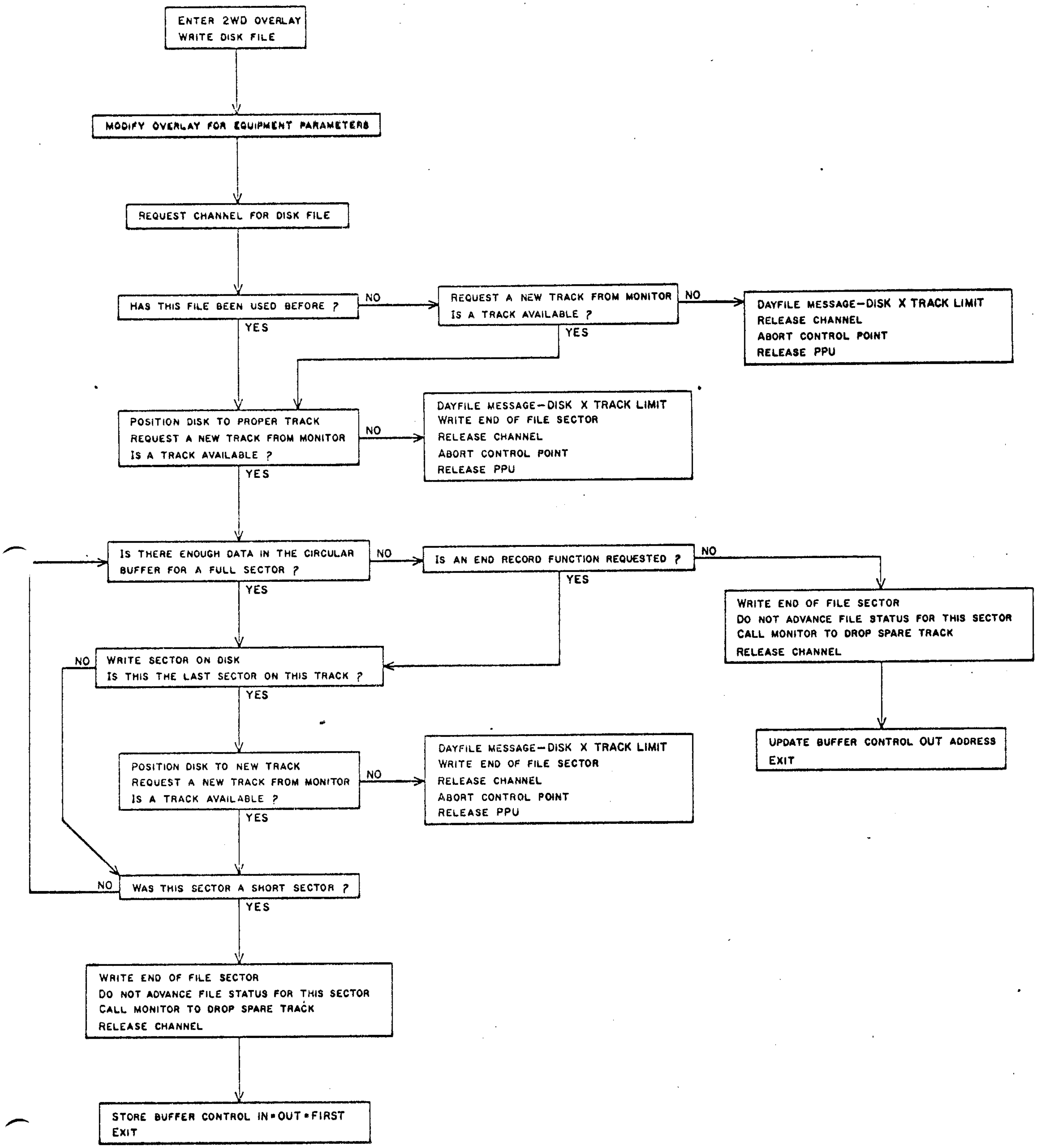
2BD then searches each word in the buffer from OUT - 1 down to FIRST until a word with a zero low-order byte is found, indicating the end of a logical BCD record. When the end of the record is found, 2BD updates the IN and OUT pointers in the calling program's argument list, and returns control to CIO. OUT now points to the first word following the end of the logical record. If no zero low-order byte was found, then 2BD backspaces two sectors and reads one, and then repeats the buffer search.

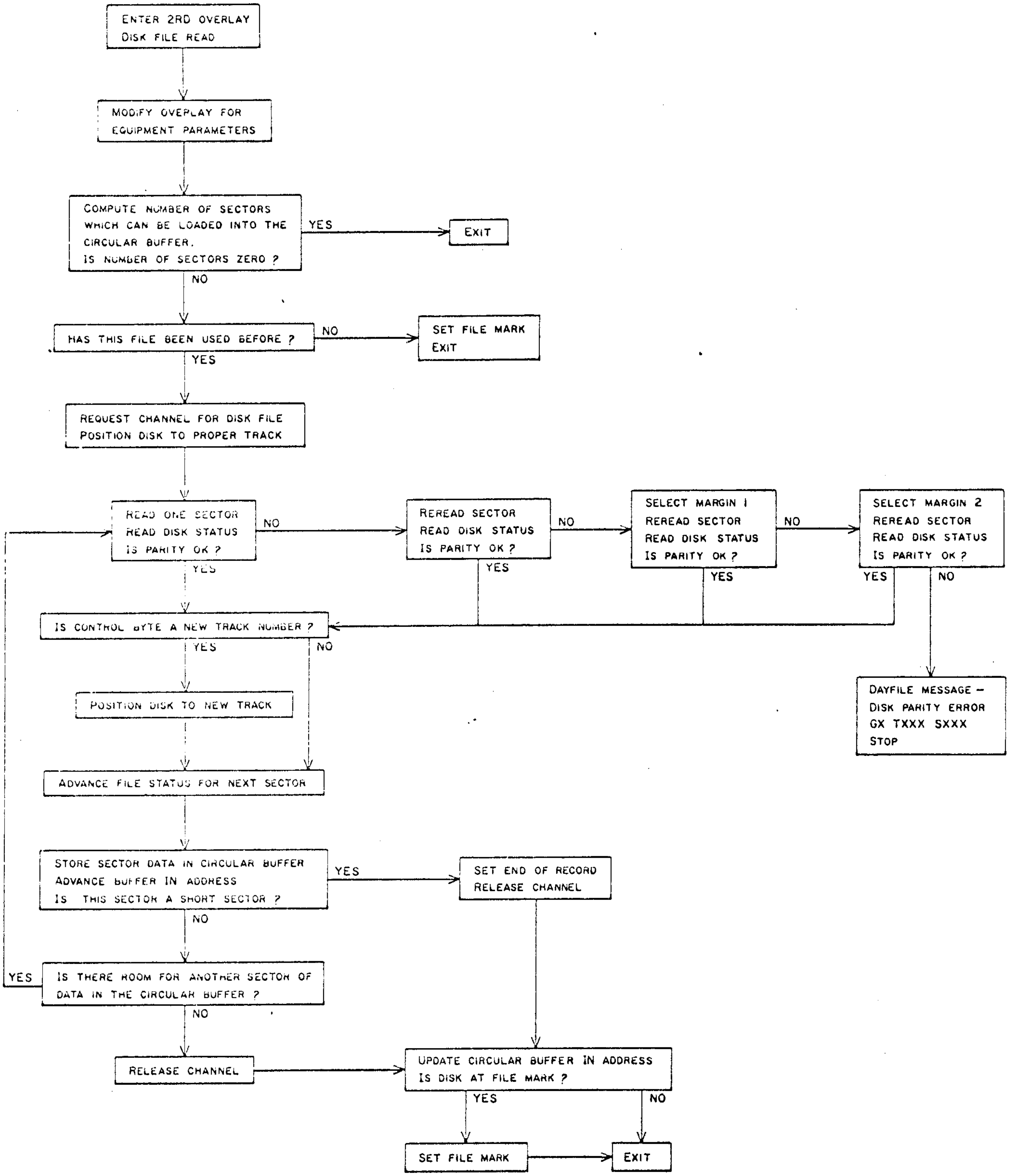
The Drop Track Overlay, 2DT

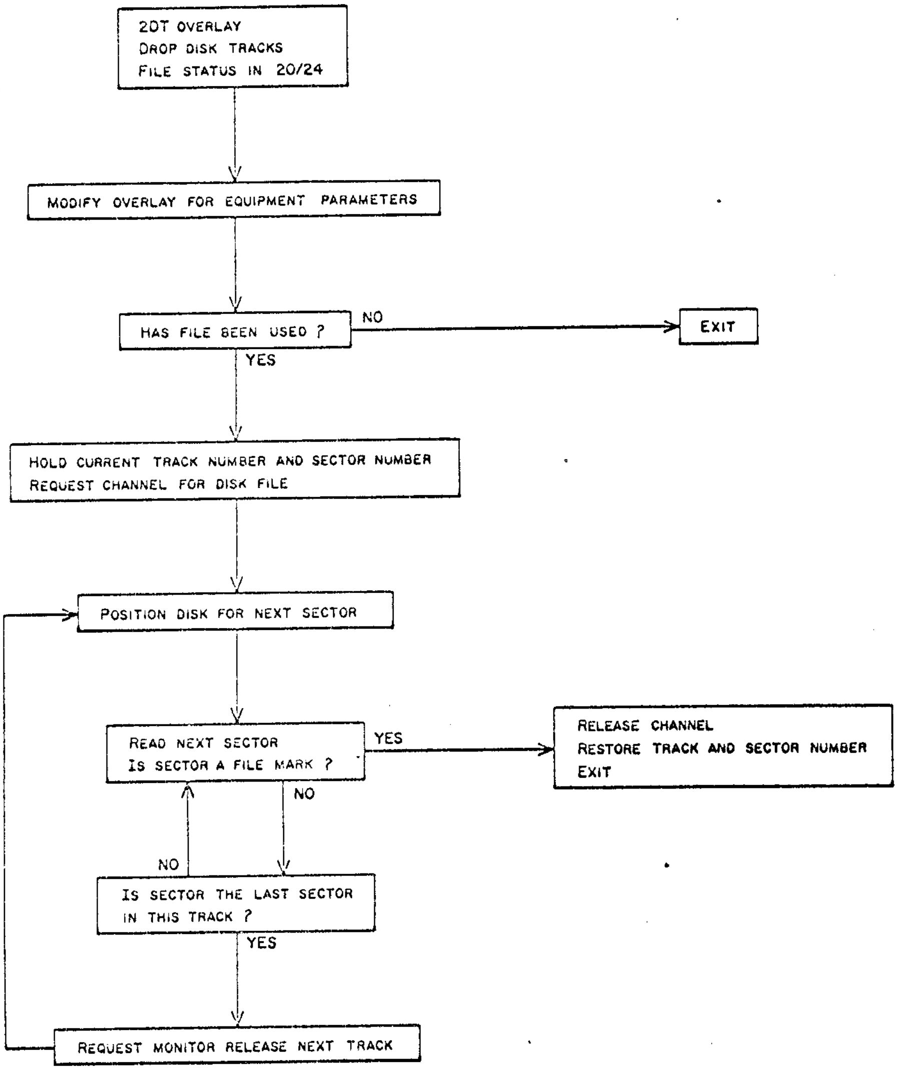
When CIO receives a disk write request, it first calls the 2BP overlay to check the legality of the buffer parameters and to search the FNT for the file name. CIO then reads the EST entry for this file, and examines the buffer status in byte 5. If the buffer status indicates that the last operation performed on this file was a read operation, then an overlay, 2DT, is called to drop the subsequent portion of the file. In effect, then, if some part of a file is read and it is then decided to write to that file, the remainder of the file is erased.

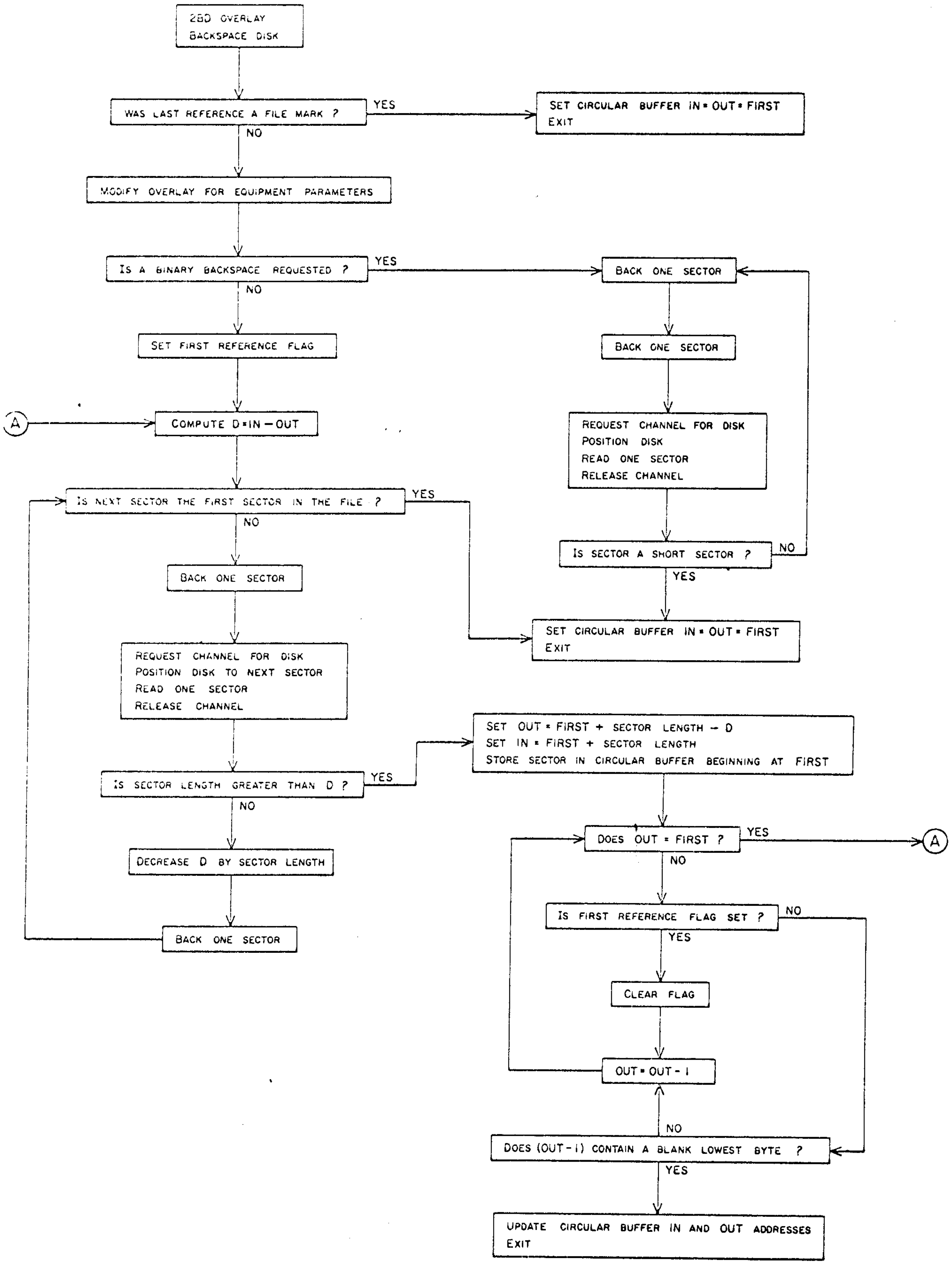
The flow chart for the 2DT overlay is shown on page A-3 of the attached flow charts. The routine picks up the Current Track byte and Current Sector byte from the FST entry for the file, and reads the sector at this address. If this sector is a file mark, 2DT returns control to CIO. If control byte one of this sector contains a half track address, 2DT requests MTR to drop this half track reservation. MTR then clears the bit in the Track Reservation Table corresponding to this half track address. 2DT positions the disk to this half track address and begins reading sectors until a file mark is found or the end of the half track is reached. The process of reading and dropping half tracks continues until the end of the file is reached.

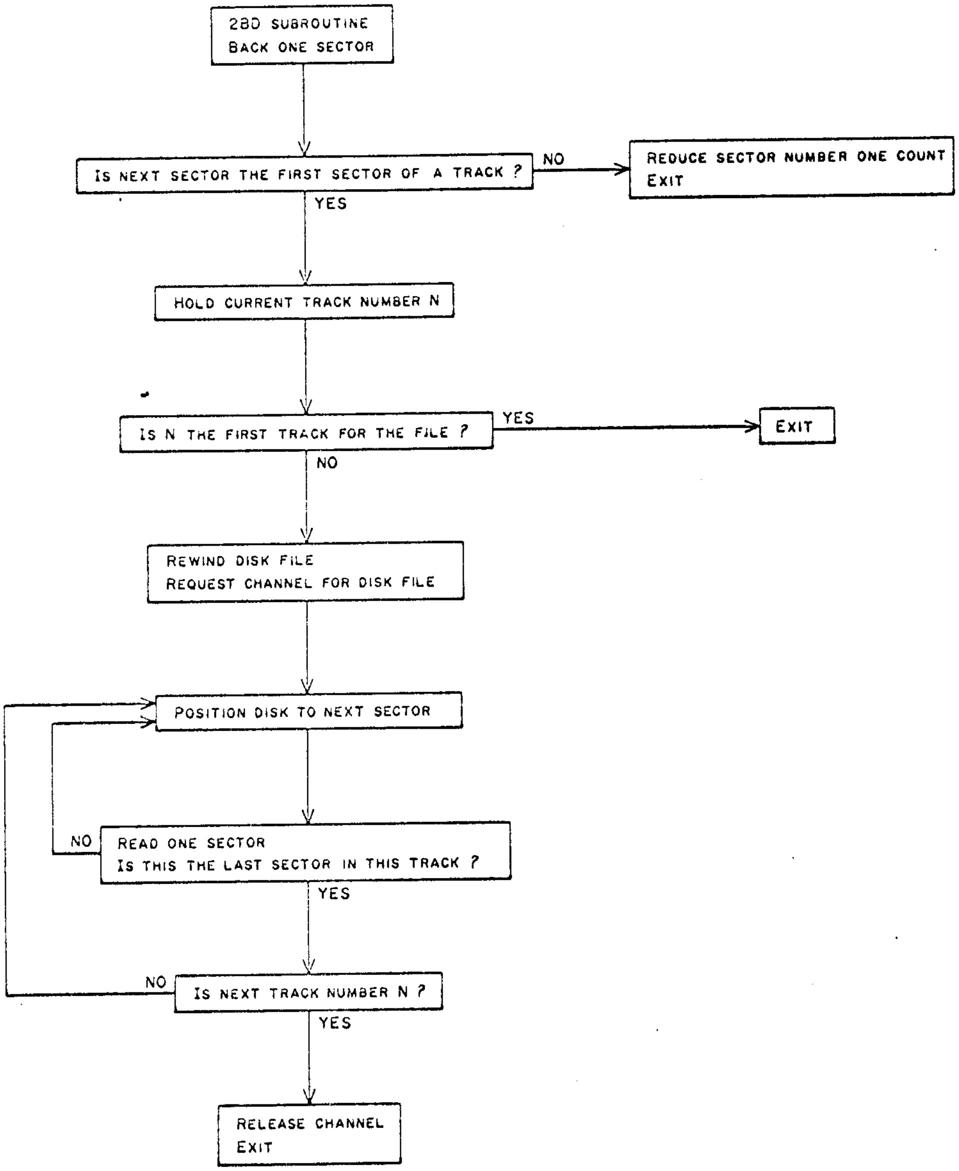
At the end of a job, all local files associated with the job are dropped. For disk files, a process similar to that described above is required to release half track reservations. This is performed for 1AJ by the 2DF overlay. 2DF differs from 2DT in that 2DF drops files assigned to other equipment as well as those assigned to the disk, and 2DF drops all the half tracks reserved by a file, not just those following the half track specified in the Current Track byte of the FST entry. 2DF is also called by 1DJ and 1TD when printing files or writing files on tape.


ENTER 2WD OVERLAY
WRITE DISK FILE
MODIFY OVERLAY FOR EQUIPMENT PARAMETERS
REQUEST CHANNEL FOR DISK FILE
HAS THIS FILE BEEN USED BEFORE ?
NO
REQUEST A NEW TRACK FROM MONITOR
IS A TRACK AVAILABLE ?
NO
DAYFILE MESSAGE—DISK X TRACK LIMIT
RELEASE CHANNEL
ABORT CONTROL POINT
RELEASE PPU
YES
YES
POSITION DISK TO PROPER TRACK
REQUEST A NEW TRACK FROM MONITOR
IS A TRACK AVAILABLE ?
NO
DAYFILE MESSAGE—DISK X TRACK LIMIT
WRITE END OF FILE SECTOR
RELEASE CHANNEL
ABORT CONTROL POINT
RELEASE PPU
YES
IS THERE ENOUGH DATA IN THE CIRCULAR
BUFFER FOR A FULL SECTOR ?
NO
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
YES
NO
WRITE SECTOR ON DISK
IS THIS THE LAST SECTOR ON THIS TRACK ?
YES
UPDATE BUFFER CONTROL OUT ADDRESS
EXIT
POSITION DISK TO NEW TRACK
REQUEST A NEW TRACK FROM MONITOR
IS A TRACK AVAILABLE ?
NO
DAYFILE MESSAGE—DISK X TRACK LIMIT
WRITE END OF FILE SECTOR
RELEASE CHANNEL
ABORT CONTROL POINT
RELEASE PPU
YES
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 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


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


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
DECREASE D BY SECTOR LENGTH
NO
IS FIRST REFERENCE FLAG SET ?
NO
BACK ONE SECTOR
YES
CLEAR FLAG
OUT = OUT - 1
NO
DOES (OUT - 1) CONTAIN A BLANK LOWEST BYTE ?
YES
UPDATE CIRCULAR BUFFER IN AND OUT ADDRESSES
EXIT


28D 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