File Number 1410-30
Form C28-0541-1

IBM Systems Reference Library

IBM 1410 Input / Output Control System

Programming Systems Analysis Guide

This manual provides detailed information concerning the internal logic of the IBM 1410 Input/Output Control System. The manual is addressed to technical personnel responsible for analyzing or modifying the program. The charts and detailed descriptions herein are based on Version 3, Level 0 of I/O 926.

Preface

The reader of this manual should have a basic knowledge of the IBM 1410 Autocoder language. Familiarity with the information contained in the following manuals is necessary to understand the material contained in this manual:

IBM 1410 Input/Output Control System for Card and Tape Systems, Form C28-0334
IBM 1410 Autocoder, Form C28-0309
IBM 1410 Principles of Operation, Form A22-0526

MINOR REVISION (January 1964)
This edition is a minor revision of the preceding edition, Form C28-0541, but does not render that publication obsolete.

Revisions to the text are indicated by a vertical line to the left of the change; revised illustrations are denoted by the symbol (•) to the left of the Figure caption.

Copies of this and other IBM publications can be obtained through IBM Branch Offices.
Address comments concerning the contents of this publication to:
IBM Corporation, Programming Systems Publications, Dept. D91, PO Box 390 Poughkeepsie, N. Y., 12602

Contents

List of Illustrations

The Input/Output Control System for the IBM 1410 Data Processing System is comprised of program routines written by the IBM Programming Systems to provide users with efficient, pretested routines for reading and writing card and tape records.

Programming of input and output routines that handle records efficiently is difficult. The routines used in IOCS have been found through experience to be efficient. By using this system in all programs, standard input and output routines are provided. Such routines simplify and standardize console operations.

IOCS provides the following features while satisfying the requirements for reduced programming, efficient routines, standardization, and elimination of input/output programming errors:

1. Reading and writing of data records simultaneously with processing.
2. Macro-instructions that handle records sequentially, even though they are in blocked form on an input tape, or are to be written in blocked form on an output tape.
3. Checks for proper mounting of input tapes and aids in the checking of each tape used. By the use of label records, each reel of tape may be identified and checked before being used in the program.
4. Routines for processing unit records. Unit records may be read, punched, or printed on-line using macros.
5. Error routines for tape and unit records, that correct errors whenever possible.

The functions provided by IOCS are incorporated into the user's program during assembly by IBM 1410 Autocoder. Flexibility is given the user by allowing him to specify in free form a variety of program parameters with a minimum number of cards. Only the necessary coding is generated to reduce storage requirements.

Whenever programming or machine errors occur during an input or output operation, they are often difficult to diagnose because of the complexity of these operations. If IOCS is being used as a standard input/output routine, a thorough understanding of how it operates becomes desirable in order to be able to diagnose quickly any difficulties in machine operation that might occur in this area.

The manual describes general as well as detailed flow charts to aid in understanding the operation of IOCS. However, no attempt is made to describe IOCS checkpoint and restart, real time, or disk applications.

Two conventions are used throughout this manual in reference to IOCS labels. The dollar sign ($) is used in lieu of IOCS, as the first four label characters e.g., $ENTRY instead of IOCSENTRY. The hyphen (-) is used in labels that have variable prefixes. One hyphen is used in lieu of the channel number (e.g., $CS-SFS), and two hyphens in lieu of a file prefix (e.g., $--FULL).

Tape File and Channel Schedulers, Interrupt

Scheduling is that part of IOCS that controls the manner in which I/O needs for files are serviced. For tape files, the scheduling is performed by file schedulers in correlation with one or more channel schedulers.

File Scheduling

The specification of a tape file in the DTF, at compilation time, causes the generation of one file scheduler. The latter serves as a common subroutine for all GET or PUT macros referring to the file. The file scheduler is entered from a GET to make the next block of logical records available to a GET macro when the current area is empty, i.e., all logical records in the current block have been processed. For a PUT macro, the file scheduler makes the next block area available for processing when the current area is full, i.e., has been filled with logical records. The I/O instructions to read a block of records into storage or write a block out are initiated in the file scheduler.

The file scheduler coding generated varies according to the application specified by the DTF for the file. The file type (e.g., output tape), the data characteristics (e.g., variable-length, blocked records), and the optional features (e.g., the specification of two buffer areas in conjunction with the overlap and priority special features) all influence the coding generated.

BUFFERED FILE SCHEDULER, TWO AREAS

Use of two areas for a file allows a look-ahead approach that enables the overlapping of processing and I/O. Case Studies 1, 2, and 3, on Chart A, illustrate how this is done. In Chart A, the number associated with a block is a step number. In the case study descriptions, the execution of a step is indicated by *(n)*, where *n* stands for the step number; reference to another step by step n.

Case Study 1: One file only on channel 1 (file A). An I/O operation has been started in area 2 and processing is starting in area 1 *(1)*.

While area 1 is being processed, the area 2 I/O operation terminates *(2)*, and an interrupt to check it occurs *(3)*. Since no new I/O operation can be started *(4)*, control returns to complete the processing in area 1 *(6)*. Then, the file scheduler is entered. A test indicates area 2 is available *(7)*, the need for an I/O in area 1 is flagged *(12)*. The channel is free *(13)* and, since an I/O operation is to be started *(14)*, it is started for area 1 *(15)*. Processing is then resumed with area 2 *(16)*. Note that I/O and processing were not overlapped during step 6.

PENDING SWITCH

Every two-area file scheduler has a pending switch. The pending switch is set on to indicate that a tape operation is to be initiated in an area and it stays on until the needed I/O operation is started, completed, and checked. In case study 1, the pending switch is set on at step 1, turned off at step 3, and turned back on at step 12.

Case Study 2: Same as case study 1, except there is another file on the channel (file B). The pending switch for file B is on.

At step 3, the channel is clear so the pending switches are "interrogated" *(4)*. Since the switch for file B is on, a tape operation for file B is started *(5)*. Assuming the channel is still busy at step 13, steps 14 and 15 are omitted. Note how processing and I/O are overlapped throughout the sequence traced.

FORCING SITUATION

A forcing situation for a two-area file occurs when entry is made to the file scheduler and another area is not available. Two possibilities, with resultant consequences, can exist:

1. The channel is busy with another file's I/O operation:
   a. The channel is cleared.
   b. An I/O operation for the forced file is started.
   c. The channel is cleared.
2. The channel is busy with this file's I/O operation:
   a. The channel is cleared.

Note that for a one-area file, every entry to the file scheduler is a forcing situation.

Case Study 3: Same as case study 2, except that the I/O operation for file A has not terminated by the time step 7 is entered. Steps 2 through 5 consequently do not exist.

Since the test of the pending switch at step 7 determines that file A does not have an available area, it is a force situation. A force loop is now entered to wait till the I/O operation terminates *(8)*.

The I/O is then checked *(9)*, completing the clear channel operation. The pending switch is tested again *(10)*, and is now found to be off. Therefore, the force situation comes to an end. The channel is not busy at step 13, so step 14 is entered, but the I/O started at step 15 may be either for file A or B (since both pending switches are on). Note that the pending switch tested in step 10 would be on if the operation cleared on the channel was for another file; an I/O operation for the forced file would be started at step 11, and the force loop would be re-entered.

PENDING NETWORK AND PRIORITY

The pending switches for all two-area files on a channel are linked together. This aggregate is called the pending switch network for the channel. The linkage from one pending switch to another is via their off-status branch addresses. For the pending switch below:

Label	*Operation*	*Operand*	*Explanation*
	NOP		WM over BXPA = pending on
	BXPA	XXXXX	No WM over BXPA = pending off

XXXXX represents the address of the NOP in another pending switch. The next sequential instruction following the pending switch is the I/O instruction for the file.

The pending switches in the network are linked in high-to-low (0-9) relative priority for the respective files on the channel. When the network is entered at the top, control passes down the network via the off-branches until a pending switch is found in on status. Control then drops through the switch to the tape operation. The pending network is ordered by the priority assignment routine during an OPEN operation for all two-area files named in the OPEN (or for files already OPEN), unless PRIORITY ASSEMBLE was specified, in which case the assignment was done at compilation time.

In the case studies, the pending switch network is entered in step 4 and in step 14. Note that the I/O started in step 15 of case 3 depends on the priority assigned files A and B.

Channel Scheduling

The channel scheduler is generated by the specification of DIOCS CHANNEL. The file scheduler uses the channel scheduler as a subroutine for two main operations. The entries and operations are:

1. Clear channel operation: This is done in a forcing situation; e.g., case study 3, steps 8 and 9. The channel must be forced clear of any unchecked I/O operation before another I/O operation can be started for the forced file.

2. Start channel operation: This is done just before the file scheduler returns control to the macro; i.e., case studies, steps 13, 14, and 15. If the channel is not busy (step 13), the pending network is entered (step 14). If there is an I/O operation to be started, it is started for the highest priority pending switch found on (step 15).

These operations are also executed, when necessary, during end of reel and macro processing of OPEN, CLOSE, etc.

INTERRUPT

Interrupt coding also links into the channel scheduler. After the operation is checked and the pending switch turned off, the channel is started again by going to the pending network (see case studies, steps 3, 4, and 5).

TWO-CHANNEL OPERATIONS

When there are two channel schedulers, the channel 1 scheduler exits into the channel 2 scheduler for non-forcing operations.

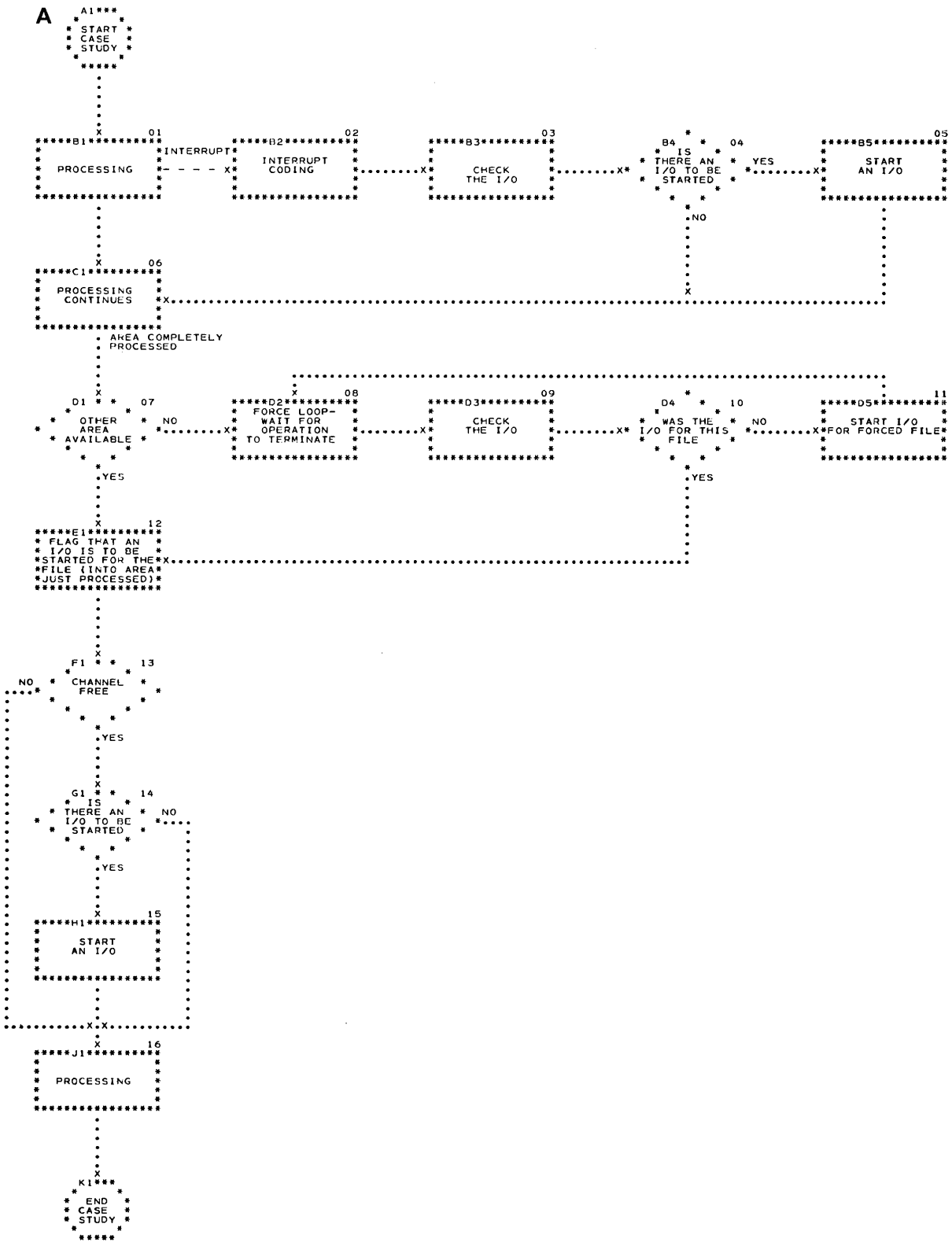


Chart A. Case Studies 1, 2, and 3

The overall processing in IOCS is covered by Charts AA and AB and by Figure 1. Chart AA is an overall chart for file and channel scheduling; Chart AB covers OPEN, CLOSE, FEORL, RDLIN, and end of reel. These two charts are described by functional operation rather than block by block. Figure 1 is an overall table that summarizes tape and unit record error procedures. The description of Figure 1 discusses these procedures more fully.

Some of the descriptions refer to the file reference address. This address is the starting location of the file reference table for the file. The reader may find it helpful to refer to this table, included as Figure 7, and described in the Programming Condition Analysis Aids section of the manual.

In the operational descriptions of Charts AA and AB, the convention (xxnn) indicates the block whose processing is being described. The convention block xxnn, or (at block xxnn) is used to refer to another block. The reader is again reminded that in a label, the convention of two hyphens (--) designates the file prefix (e.g., $--FULL) either generated by IOCS or specified by the user. One hyphen (-) designates the channel prefix (e.g., $CS-SFS). The dollar sign ($) substitutes for the first 4 characters in the label, viz., IOCS.

Scheduler Operations, Interrupt

Chart AA is an overall flow chart for file and channel scheduling. The channel scheduler, shown on columns 1-3 on Chart AA, is described for three operations: Start Channel Operation, Interrupt Operation, and Force Channel Clear Operation. The file scheduler, shown on columns 4-5 on Chart AA, is described for three operations: Two-Area Operation, Non-Force; Two-Area Operation, Force; and One-Area Operation.

Scheduling is covered on a detailed level in Charts BA, BB, BC, BD, BE, and BF.

Channel Scheduling

START CHANNEL OPERATION

This operation occurs to restart the channels, and is executed whenever appropriate in various IOCS routines. If a channel is already in operation, control does not wait for the operation to terminate. A BOL on each channel acts as a gate to further action. The following table shows the possibilities:

Channel 1	*Channel 2*	*Action*
Busy	Busy	———
Busy	Free	Start channel 2 if possible after checking the previous I/O operation, if any.
Free	Busy	Start channel 1 if possible after checking the previous I/O operation, if any.
Free	Free	Start both channels if possible after checking previous I/O operations, if any.

The operation starts by an SBR (AA01) to set the exit linkage at $INTEXT, block AA23. A test of the disable switch follows to determine if a start channel operation can be made (AA02). If the disable switch is on, control returns immediately to the original routine. Otherwise, the channel 1 scheduler is entered at $CS1ENT, block AA04, by means of a BXPA.

If channel 1 is in operation (AA04), the BOL1 branches to the channel 2 scheduler, $CS2ENT, block AA13. Otherwise, a BOPR1 tests if there is an unchecked I/O operation on the channel (AA05). If there is, the BOPR branches to the file scheduler coding block to make the check, block AA06. On return from checking, or if no check is made, control drops through the force switch which is off (AA09).

The inquiry (BIPR1) or unit record (BUPR1) request latches are serviced (not shown) if applicable and then the pending switch network is entered (AA11). If no pending switch is on, an I/O operation cannot be started, and control branches from the lowest-priority pending switch on channel 1 directly to the channel 2 scheduler at $CS2ENT, block AA13. If an I/O is started, it is for the file of highest priority whose pending switch is on. Return to the channel 1 scheduler in this case is made at CS1RET (AA12) to set the address of the status check, coding block AA06 (in the file scheduler for which the I/O operation was started), into the branch address of the BOPR1 at block AA05. If the I/O instruction was NOP'ed when executed, control remains in the file scheduler to execute the status check immediately. After a successful re-execution, the return to the channel scheduler is made to block AA09.

The channel 2 scheduler is then entered at $CS2ENT (AA13). If channel 2 is in operation, the BOL2 branches to block AA22. If channel 2 is not busy, the operation of blocks AA14 to AA15 and AA18 through AA21 is identical to the channel 1 operation already described. Control then proceeds, via block AA22, which has no effect, to $INTEXT (AA23). An unconditional branch is made as set by block AA01 and the priority alert mode is re-entered.

INTERRUPT OPERATION

The operation is similar to a start channel except that both channels cannot be busy because an I/O operation terminating on one of the channels caused the interrupt.

The operation is begun at core storage location 00101. $INTEXT, block AA23, is set with the contents of the B-register less 6, and the 1411 status is saved if required (AA03). The channel 1 scheduler is then entered at $CS1ENT (AA04).

After channel scheduling operations are finished, the 1411 machine status is restored if it was saved (AA22). Control then re-enters the priority alert mode (AA23) and branches as set by block AA03 to the instruction at which the interrupt occurred.

FORCE CHANNEL CLEAR OPERATION

This operation can be executed for either channel. It forces an I/O operation in progress to terminate before it is checked. It does not restart the channel or affect the other channel. A channel is forced clear before issuing an I/O command in a 1-area file scheduler, in all unit record schedulers, and in a 2-area file scheduler when a forcing situation exists.

The operation for channel 1 starts by an SBR (AA07) to set the exit linkage in the channel 1 force exit, $CS1SFX block AA10. The force switch at block AA09 is set on (AA08). A sequence, BOL1 to $CS1PR, BOPR1 to $CS1PR, is entered (AA09). If channel 1 is free and clear, control drops through the sequence, the force switch is reset off (AA10), and control returns to the proper routine as set by block AA07.

If channel 1 is free but not clear, control drops through the BOL1 to the BOPR1 which branches to $CS1PR, block AA05. At $CS1PR another BOPR1 branches to the file scheduler coding block to check the operation (AA06). After checking, the force switch is on, so control returns to the BOL, BOPR sequence (AA09) which is now dropped through since the channel is free and clear.

If channel 1 is busy, the BOL1 branches to $CS1PR, block AA05. There control drops through the BOPR1 to block AA09, unless the overlap request latch for channel 1 was just set. At AA09, control returns to the BOL1, since the force switch is on. The loop is repeated until the overlap request latch is set. The channel is then free but not clear, and further processing is as previously described.

Processing for a force channel 2 clear operation, blocks AA16 through AA19 and AA14 to AA15, follows logic identical to that for channel 1.

In a bootstrap force operation, entry is made at block AA08, to utilize the already-established force exit linkage at block AA10, after the bootstrap operation is forced clear.

File Scheduler Operations

Entry to the file scheduler is made from a macro whenever all logical records have been processed in an area. The main function of the file scheduler is to make another area available and prepare it for the macro.

TWO-AREA OPERATION, NON-FORCE

The file scheduler exit, block AA38, is set for return to the macro. Housekeeping is performed if needed (AA31), e.g., an input file has 1 added to its block count. The pending switch is tested to see if the other area is available (AA32). If it is (pending switch off), control branches to $--PA. At $--PA, housekeeping is performed to enable the macro to process the new area and the pending switch is set on to signify the need for an I/O operation in the just-processed area (AA33). Further housekeeping is performed if necessary, such as resetting the area limits for blocked records (AA34). $--TRIG is normally (AA35) a branch to $ENTRY, block AA01, to start the I/O operation for this or a higher-priority file (AA36). On return from starting an I/O operation, further housekeeping is done such as adding 1 to the block count for an output file (AA37). Control then exits the file scheduler to the macro that caused entry (AA38).

Note that $--TRIG is also used as a pivot for exceptional conditions. In these cases, $--TRIG is set by some other IOCS routine. These conditions include WLR processing for input files (if specified), priming for input files, and linkage to the end-of-reel routine for all files. In WLR processing, $--TRIG is reset to normal. If the user decides to accept the record, control returns to $--TRIG. If the record is rejected, another tape record is read to replace it by returning to block AA32 to start a force operation. In a prime operation, $--TRIG is reset to normal and control returns to AA32 to begin a force operation. The setting of $--TRIG to branch to WLR or end-of-reel processing was done in the IOCS error routine; the setting of $--TRIG to branch to prime was done during open procedures either from an OPEN macro or end-of-reel processing.

TWO-AREA OPERATION, FORCE

If the test at block AA32 indicates that the other area is not available (pending switch on), a force situation exists. A BXPA to the appropriate channel scheduler force entry, block AA07 or AA16, is made to force the channel clear (AA39). After the channel is clear, $INTEXT, block AA23, is set to go to the bootstrap force entry in the channel scheduler at block AA08 or AA17, in case a bootstrap force is needed (AA40). The pending switch is tested again to see if the area has been made available (AA41). There are two possibilities:

1. Pending switch off: The I/O operation cleared on the channel was for this file. The force situation no longer exists, and control branches to $--PA, block AA33.

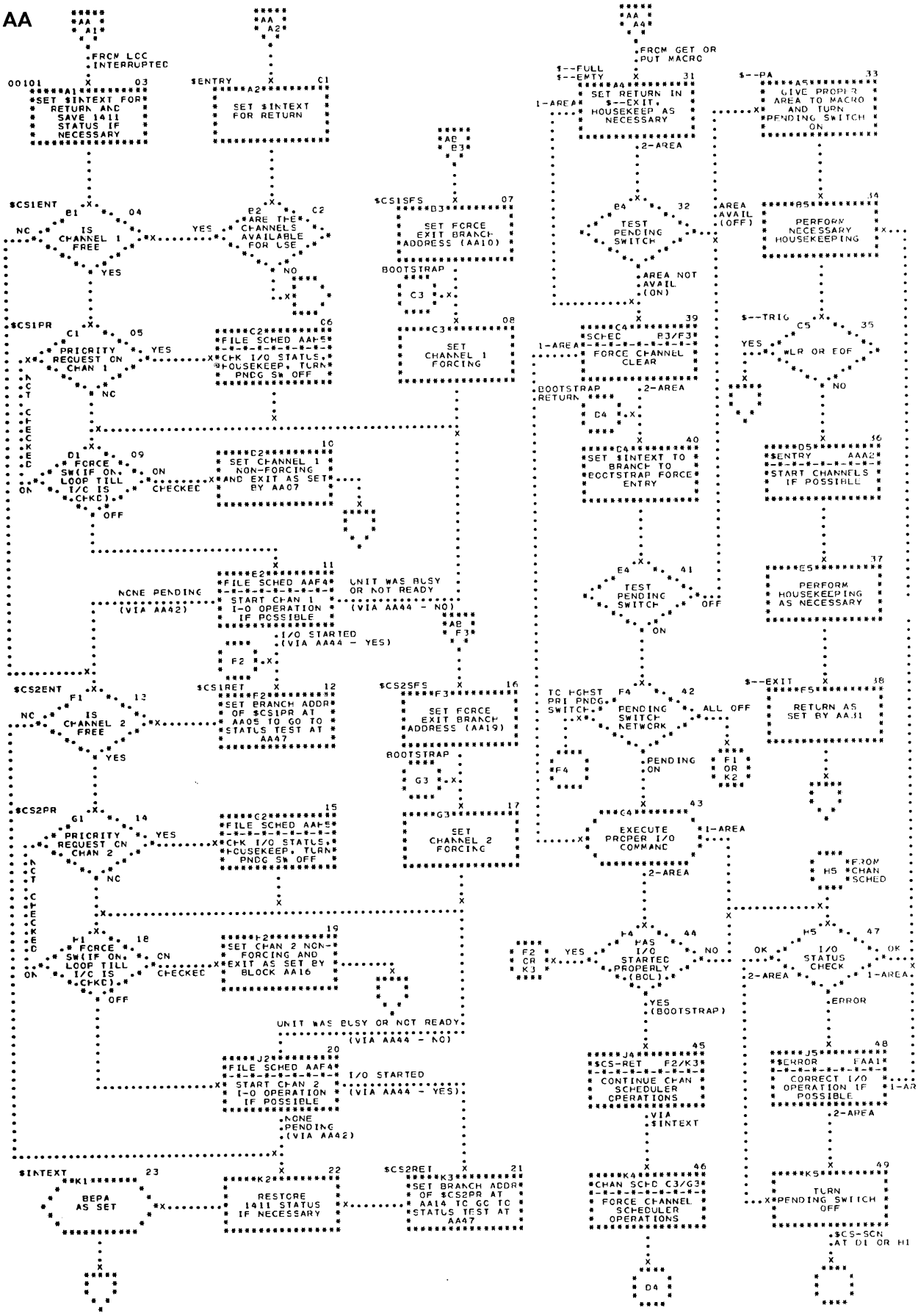
2. Pending switch on: The I/O operation cleared on the channel was for another file, and so the bootstrap force is necessary.

Control drops through the pending switch which is a NOP (AA42) to execute the I/O instruction (AA43). If the I/O operation was started, the BOL- branches to $-RET in the channel scheduler to set the status test linkage (AA44). Channel scheduler operations eventually exit at $INTEXT (block AA23) which has been set (by block AA40) to re-enter the channel force routine at the bootstrap entry of AA08 or AA17. The channel is forced clear of the I/O operation just started, control returning to block AA40 via the already established force exit linkage (AA45). When the pending switch is tested, it is now off (AA41), the force situation no longer exists, and control branches to $--PA, block AA33.

If the I/O at block AA43 was not started, control drops through the BOL- to make the status check and turn the pending switch off in-line (AA47 through AA49). The channel scheduler is then entered with control going via its exit, $INTEXT, block AA23, to the bootstrap force entry as described before.

ONE-AREA OPERATION

The file scheduler exit, block AA38, is set for return to the macro and any necessary housekeeping is performed (AA31). The channel is forced clear (AA39) and the I/O operation is executed (AA43). An SER or SFR follows the latter when variable-length records with an WLR check are specified. The status test is performed (AA47). This forces the operation to terminate. If the tape error routine is entered, corrective procedures are performed with or without manual intervention (AA48). After any necessary area-control housekeeping is performed (AA34), $--TRIG branches to $ENTRY to restart the channel (AA36, AA37). On return, further housekeeping is performed, such as checking Form 4 records (AA37). Return to the macro is made (AA38).



- Chart AA. Channel and File Schedulers — Overall

OPEN, CLOSE, FEORL, RDLIN, and End of Reel

Introduction

The routines generated to process OPEN and CLOSE macros depend on the specifications in the DIOCS card packet. The inclusion of TAPE as an IODEVICE generates the bulk of the coding. FEORL, RDLIN, and end-of-reel are exclusively for tape.

The DIOCS routines generated form one logical unit which is shown as Chart AB. Much of the coding is shared because of the similarity of the functions performed. For example, the part of the end-of-reel processing that is concerned with opening a new reel of tape is essentially the same as the open procedures performed for an OPEN macro. Detailed charts for all the operations are included as Charts DA, DB, DC, DD, DE, DF, and DG. Subroutine blocks have been used in Chart AB to show its relation to the detailed charts.

MACRO FORMAT

The out-of-line block of coding executed for a macro is a subroutine to the macro. The macro itself is in-line with respect to the user's coding. It consists of linkage to the DIOCS routines and a calling sequence listing the files to be processed by their file reference addresses (file names). Each address is preceded by a check character which is a code to designate the type of macro. The last file named in the calling sequence is always followed by a termination character of J and a termination address. The files named in an OPEN or CLOSE macro may be unit record or tape. The execution of the coding for a particular macro will be called an operation. An OPEN macro is given as an example. The source statement, "OPEN FILEA, FILEB, FILEC," would be compiled into the one-for-one statements:

Label	*Operation*	*Operand*	*Explanation*
	B	$CLOP	Linkage to DIOCS routines.
	C	FILEA	C is the check character for an OPEN operation. FILEA, etc., are the file reference addresses or file names.
	C	FILEB	
	C	FILEC	
	B	$ENTRY	Terminal character and address.

The OPEN operation starts by a branch to $CLOP. After all files are opened in calling sequence order, control returns to the branch to $ENTRY instruction.

END OF REEL

The out-of-line block of coding executed for end-of-reel is a subroutine to the file scheduler. The exit linkage from the file scheduler is set up by the tape error routine after sensing a tape mark (input) or reflective spot (output).

Descriptions of Processing

PRELIMINARY PROCESSING, BLOCK AB01

A description of the initialization which precedes file-by-file processing follows:

1. The first file to be processed is identified. This is done by setting up a pointer to the beginning of the macro calling sequence.
2. The program leaves the priority alert mode of operation.
3. The channels are cleared of all unchecked two-area tape operations that have been previously initiated.
4. IOCS is set so that if an I/O macro is executed in a user routine, its execution cannot lead to the resumption of normal channel operations (via $ENTRY). The interrogation of pending switches (two-area tape files) and of priority request latches (inquiry, real-time, etc.) are bypassed. See discussion of disable switch in Start Channel Operation, Chart AA.
5. The contents of index register 15 are saved so that IOCS can use this index register for file processing.

INITIAL FILE-BY-FILE PROCESSING, BLOCKS AB02-AB03

File-by-file processing begins (or continues) by moving the first (or next) segment of the macro sequence to a work area (AB02). A segment consists of the check character addressed by the pointer (see step 1 under "Preliminary Processing"), file reference address, and the check character for the next file. The file reference address is moved into index register 15 for processing the current file; the pointer is then incremented by 6 to prepare for the next file. A test is then made to determine if it is a RDLIN macro. If it is, control branches to block AB24.

Otherwise, a table look-up is executed by file type in a table of routine linkages (AB03). For each file type, there is a sequence of two addresses; one for OPEN, the other for CLOSE (or FEORL) operations. The bit structure of the check character in the calling sequence work area is examined to determine which address to use.

PRIORITY ASSIGNMENT, BLOCKS AB04-AB07

The priority assignment routine, if generated, is executed only during an OPEN operation. The considerations for all DIOCS configurations are as follows:

A. *Non-Overlay:* The routine remains in storage at all times, unless overlayed by the user in conjunction with the use of the origin option. The routine is entered and executed at block AB07 only for a two-area file (file type = 2), as provided by the tape sequence in the routine linkage table. (For a one-area file [file type = 1], the tape sequence causes control to go directly to open procedures, block AB08.) The pending switch address and DTF-specified priority for the file are entered into a table which contains such addresses

and priorities for all files previously opened. The table is sorted by relative priority on each channel. The pending switch for the file is then inserted into the pending-switch network on the respective channel. A previously opened file is eliminated from the table before assignment begins. Control then branches to tape open procedures, block AB08.

B. *Overlay (before the Second IOCS load):* A dummy sequence in the routine linkage table forces control to pass through the priority assignment routine for every file type. A test is made to determine if it is a two-area tape file (AB04). If it is, the file is processed in the manner described earlier (see entry A). A test is then made in the calling sequence work area to determine whether there is another file to process (AB05). If there is, control returns to block AB02. Otherwise (the J terminal character was encountered) the macro pointer is reset to point to the first file in the calling sequence. The dummy sequence in the routine linkage table is replaced by the tape sequence described under entry C. Control then branches to the load program at core location 00281. The second IOCS load is brought in to overlay the priority assignment routine (AB06). The load execute is to $EXIT, block AB02, to resume normal file-by-file processing.

C. *Overlay (after the Second IOCS load):* The priority assignment routine has been overlayed. Every tape file named in the OPEN is processed according to the new tape sequence in the routine linkage table. Control proceeds to block AB08 to start open procedures.

D. *Assemble:* Priority assignment was accomplished at compile time. The tape sequence for an OPEN is identical to that described under entry C.

TAPE OPEN PROCEDURES, BLOCKS AB08-AB12

Open procedures begin (AB08) by moving the file reference address for the file into a table which is arranged by file identification. The table is used by the error routine. Housekeeping for proper operation of the file scheduler is then executed.

The rewind procedure for the file is executed (AB09) preparatory to header label procedures. If it is a standard label file, the header label is processed according to the DTF options specified by the user (AB10). These procedures, for input and output files, are:

Input File: The label is read into the IOCS label area. The label is checked completely, partially, or not at all, depending on the CHECKLABEL entry of the DTF.

Output File: When a retention check is to be made, the label is read into the IOCS label area and checked before IOCS starts building the new label in the label area. After the fields specified by the DTF have been moved in, the user may modify the label information, or add to it, by using exit 4. If the user wants to check the label and build the new label himself, he can bypass both IOCS coding blocks by using exit 3; however, he must read the label himself. Finally, the tape is rewound and the contents of the IOCS label area are written on the tape.

Input file processing continues (AB11) from AB09 or AB10 to exit 7 procedures. Exit 7 may be used to check a non-standard header label or a label in addition to a standard label; however, the user must read the label himself. After exit 7 procedures, IOCS bypasses a tape mark, if required (TM specified on the CHECKLABEL entry in the DTF).

Output file processing continues (AB12) from AB09 or AB10 to exit 5 procedures. Exit 5 may be used for checking and creating a non-standard header label or creating a label in addition to a standard label; however, the user must write the label himself. After exit 5 procedures, IOCS writes a tape mark, if required (TM specified on the CHECKLABEL entry in the DTF).

After handling exits 5 or 7, control branches to the final tape procedures at block AB13 for all files.

FINAL TAPE PROCEDURES, BLOCK AB13

If required, a checkpoint identifier record, followed by the checkpoint record, is written on the DIOCS-designated tape. If the file has a pending switch (two-area only), it is turned off. Control then passes to the routine that tests if there is another file to process, block AB14.

COMMON TEST AND EXIT PROCEDURES, BLOCKS AB14-AB17

A test is made in the calling sequence work area to determine whether there is another file to process (AB14). If there is, control branches to block AB02. Otherwise (the J terminal character is sensed), file-by-file processing ends and the following final housekeeping is performed (AB15):

1. The address of the J character in the macro calling sequence (not the one in the work area) is set up as exit linkage.
2. The user's contents of index register 15 are restored.
3. IOCS is reset to permit the resumption of channel scheduler operations (via $ENTRY).

In a macro operation, the return is to the instruction, branch to $ENTRY, to restart the channels (AB16), and re-enter the priority alert mode.

In an end-of-reel operation, the place of return is to the file scheduler via one of two exceptional condition vectors (AB17). The particular vector used depends on whether it is an input or output file; the branch address of the vector depends on whether it is a one-area or two-area file (see Figure 7 or Chart DF for details). The file scheduler restarts the channels (preceded by a priming operation for an input file).

BEGIN CLOSE PROCEDURES, BLOCK AB18

Processing begins by determining if the current file named in the CLOSE or FEORL macro is a fixed, blocked, output file. If it is, and if there is a partially filled block waiting to be written, the block is padded with blanks or the DTF-specified padding character, and written on the output tape. If it is a CLOSE operation, the internal reel sequence counter is set to 0. Then both CLOSE and FEORL procedures join end-of-reel processing at block AB22 (input) or block AB21 (output).

END-OF-REEL PROCEDURES, BLOCKS AB19-AB22

End-of-reel processing is entered from the file scheduler via the pivot linkage set up by the error routine when the previous tape operation for the file was checked and an EOF condition was sensed.

To facilitate processing, a single-file macro operation is simulated (AB19). The calling sequence work area is set up with a check character of *, followed by the file reference address for the file, and the terminal character J. The file reference address is set into index register 15 after the user's contents are saved, the priority alert mode is exited, the channels are cleared, and IOCS is set to prevent normal resumption of channel operations. The *macro* pointer is set to address one of two exceptional condition vectors for exit linkage, depending on whether the file is input or output.

For an input file, trailer processing is begun (AB20). For a standard label file, after the label is read into the IOCS label area, the required internal counts are compared to the corresponding trailer counts. A discrepancy is noted by an appropriate message. After any exit 6 processing that the user may have included for trailer labels, the identifier field in the IOCS label area is tested for 1EOF. If so, control branches to the user's DTF-specified end-of-file address. For a non-standard label file, if there is a label, the user must employ exit 6 to process it and establish whether it is the last reel. The user must read the label himself. If the user establishes that it is the last reel, he must inform IOCS by moving 1EOF into the IOCS label area in order to get to his end-of-file address. If exit 6 is not used, control branches, without a test for end of file, to the user's end-of-file address.

For an output file, the coding block for trailer procedures, AB21, is shared by FEORL and CLOSE. A test is made to determine if there is a block waiting to be written (two-area file only). If so, the block, followed by a tape mark, is written. For a standard label file, processing continues by preparing the trailer label with the required count fields in the IOCS label area. Additional information may be entered by use of exit 1. After the label is written, exit 2 is provided for constructing an additional label; the user must write this label himself. For a non-standard label file, only exit 2 is provided for constructing the trailer(s). A tape mark is written on the tape if necessary.

Output processing joins input processing at block AB22. The tape is closed by executing the rewind option, and the reel sequence number is updated.

For a CLOSE operation, control branches to block AB13 to complete processing for the current file.

For an end-of-reel or FEORL operation, the new reel for the file is set up. If an alternate tape drive is specified, IOCS swaps base and alternate tapes automatically. Otherwise, IOCS enters a waiting loop after typing an identifying message so that the operator may mount the new reel on the same drive. Control then joins open procedures at block AB08 to open the new reel.

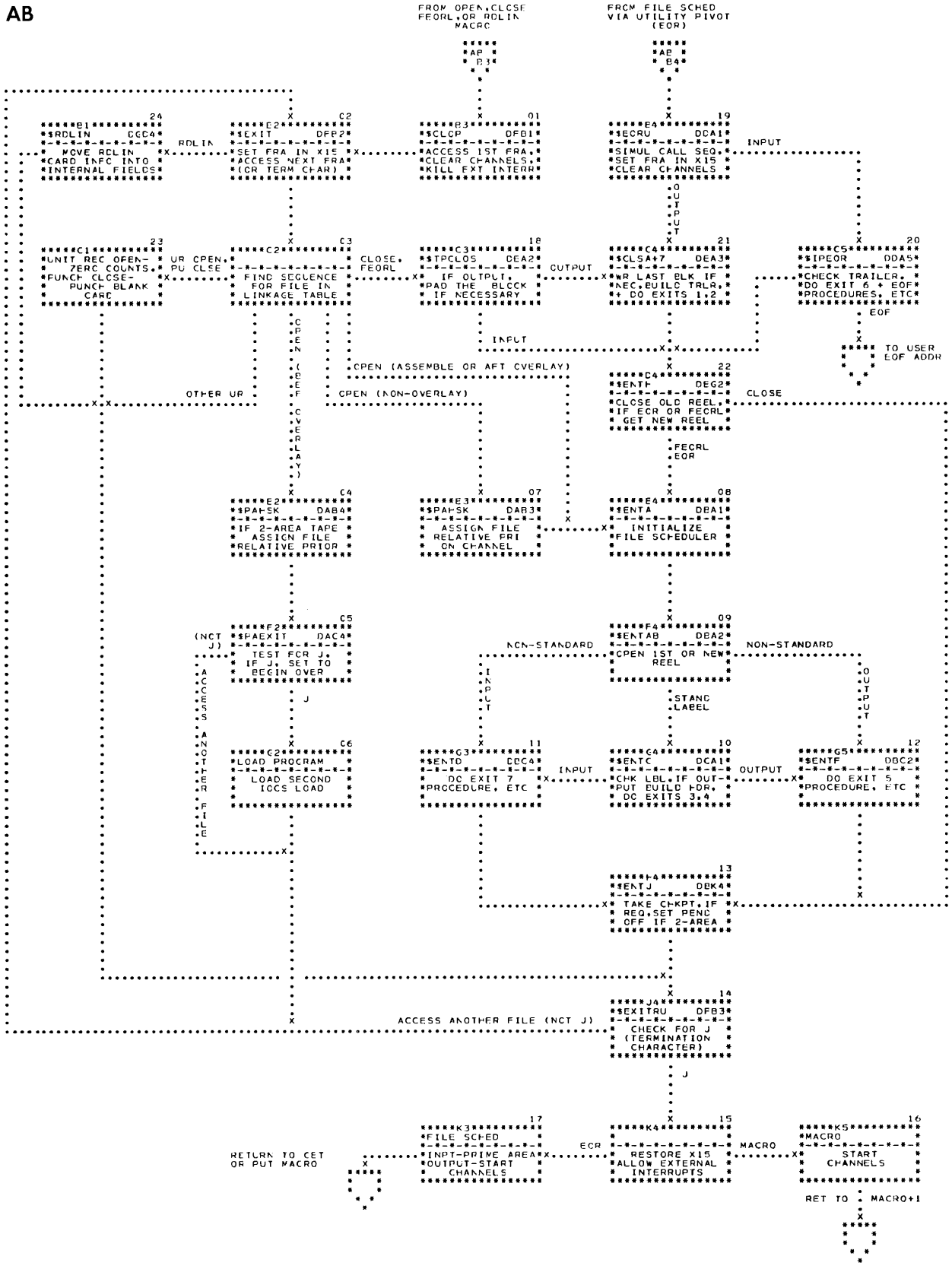
UNIT RECORD PROCESSING, BLOCK AB23

Little or no housekeeping is performed for a unit record file named in an OPEN or CLOSE macro. For an OPEN macro (assuming all of IOCS has been loaded), the block count is reset to 0. For a punch file named in a CLOSE, a blank card is punched to move the last card punched into the stacker. For a printer or reader file on a CLOSE, no processing is done.

Control then proceeds to the coding block that tests if there is another file to process, block AB14.

RDLIN PROCESSING, BLOCK AB24

The RDLIN card is read into storage and the label information is moved into the appropriate internal fields for the file. Control then proceeds to the coding block that tests if there is another RDLIN card to process, block AB14.



- Chart AB. Open, Close, EOR — Overall

General Error Operations

Two distinct error routines may be generated: one for tape if TAPE is included as an IODEVICE; the other for unit record if PUNCH, READER, or PRINTER is included. The conditions and corrective procedures for tape and unit record are summarized in the tables in Figure 1. A description of the tables follows. An overall chart for the tape error routine, Chart AC, follows Figure 1. This chart is not described. The tape error routine is shown in detail on Charts EA, EB, and EC. The unit record error routine is shown on Chart ED.

Description of Tape and Unit Record Error Routine Tables

TAPE ERROR CONDITIONS

The tape error routine is entered for two reasons: first, because of an unsuccessful I/O operation and second, because of an exceptional condition (wrong length record, first character of a record is a tape mark on read, or sensing of the reflective strip on the tape on write).

1. If the entry is due to an unsuccessful tape operation, the routine:
   a. Determines the nature of the failure.
   b. If possible, corrects the operation and returns control to the instruction immediately following the BA or BEX instruction which originally sent control to the error routine. (Note that the BA or BEX instruction just described will be referred to as the channel BA or BEX instruction in the following descriptions and charts.)
   c. If unable to correct the operation, types out an error message and enters a wait loop for operator action. An exception, explained later, to this last statement exists if autodump has been specified by a DIOCS READERROR entry with an operand of only TAPE,CU and the error is a data check on a normal read operation.
2. If the entry is due to an exceptional condition, the routine provides linkage between the file, for which it was entered, and the programmer's wrong-length-record routine or the IOCS end-of-reel routine.

The tape error routine is covered in more detail in the following description and in the left side of Figure 1. In the description, it is assumed that whenever an I/O operation is corrected through re-execution, control returns to the instruction immediately following the channel BA or BEX instruction. It is also assumed that the action taken for a described condition applies to situations where only that condition exists.

BUSY: The routine loops until the device is not busy, executes the I/O instruction and returns control to the instruction immediately following the Channel BA or BEX instruction.

NOT READY: The routine types a Not Ready message, loops until the device is made ready, executes the I/O instruction, and returns control to the instruction immediately following the Channel BA or BEX instruction.

DATA CHECK (Write): The routine attempts to correct the condition by performing the following sequence:

1. Backspace, rewrite, and check (once).
2. Backspace, skip, rewrite, and check (eighteen times).
3. Types a data check on write message (20114 DCK) and enters a wait loop for operator action. (The only possible option on a output tape error is to attempt the write operation again. This option is assumed by the error routine if the operator presses the INQUIRY REQUEST key and then the INQUIRY RELEASE key.)
4. Backspace, skip, rewrite, and check (20 times). After twenty attempts, the message and wait loop routine at step 3 is re-entered.

ZERO LENGTH RECORD (Write): The routine types a message (20117ZRL) indicating that the first character in the tape record core storage area was a group mark/word mark. A wait loop for operator action is entered. The only possible option is to proceed as if the operation were a success. This is accomplished if the operator presses the INQUIRY REQUEST key, enters the code word PROC, and presses the INQUIRY RELEASE key. Control returns to the instruction immediately following the Channel BA or BEX instruction.

DATA CHECK (Read): The routine attempts to correct the condition by executing the following sequence:

1. Backspace, re-read, and check (19 times).
2. If autodump has been specified by a DIOCS READERROR entry of only TAPE, CU and the record in error is not a label, control goes to step 5.
3. The routine types a message indicating a data check on a read operation. The message is 40119LRE if the error was on a label record, or X0113DCK, if the error was on a data record (X indicates the number of options minus one). A wait loop for operator action is entered. The options available depend, in part, on the DIOCS READERROR entry.
   A. If there is no entry, the options available are PROC, RETRY, and SKIP.
   B. If the entry is SCAN, the options available are PROC, RETRY, SKIP, and *SCAN.
   C. If the entry is TAPE, CU, no options are available.

D. If the entry is SCAN, TAPE, CU, the options available are PROC, RETRY, SKIP, *SCAN, and DUMP.

The operator selects an option by pressing the INQUIRY REQUEST key, entering the code word and pressing the INQUIRY RELEASE key. The actions initiated by the various options are:

A. If PROC is entered, the routine ignores the error and returns control to the instruction immediately following the Channel BA or BEX instruction. Processing continues as if the operation had been a success.

B. If RETRY is entered, control goes to step 4 where an attempt is made to re-read the record successfully.

C. If SKIP is entered, the routine ignores the error record, reads the next record on the tape, and returns control to the instruction immediately following the Channel BA or BEX instruction.

D. If *SCAN, is entered, the location(s) of the asterisk(s) in the error record are typed on the console printer, and the message and wait loop routine at step 3 is re-entered.

E. If DUMP is entered, the record in error is written on the DIOCS-specified dump tape and control re-enters the message and wait loop routine at step 3.

4. Backspace, re-read, and check (20 times). Control returns to the message and wait loop routine at step 3.

5. The record in error is written on the DIOCS-specified dump tape. The routine reads the next record on tape and returns control to the instruction immediately following the Channel BA or BEX instruction.

NOISE LENGTH RECORD (Read): The routine reads the next record on tape and checks it. If it too is a noise record, another read operation is performed. This sequence is repeated until ten consecutive noise records are read. At that time the message, 20118NLR, is typed and a wait loop for operator action is entered. The only possible option is to retry the read operation. This is assumed if the operator presses the INQUIRY REQUEST key and then the INQUIRY RELEASE key. Ten more attempts are made to read the record successfully. If the condition is not corrected, control re-enters the message and wait loop routine.

WRONG LENGTH RECORD (Read): The routine attempts to correct the condition by executing the following sequence:

1. Backspace, re-read, and check (10 times).
2. If unsuccessful in correcting the condition, the routine sets up linkage in the file scheduler, from which it was entered, to get to the programmer's wrong-length-record coding. Control returns to the instruction immediately following the Channel BA or BEX instruction for a two-area file or to the instruction at the file reference address (address of file name label) minus 7 for a one-area file.

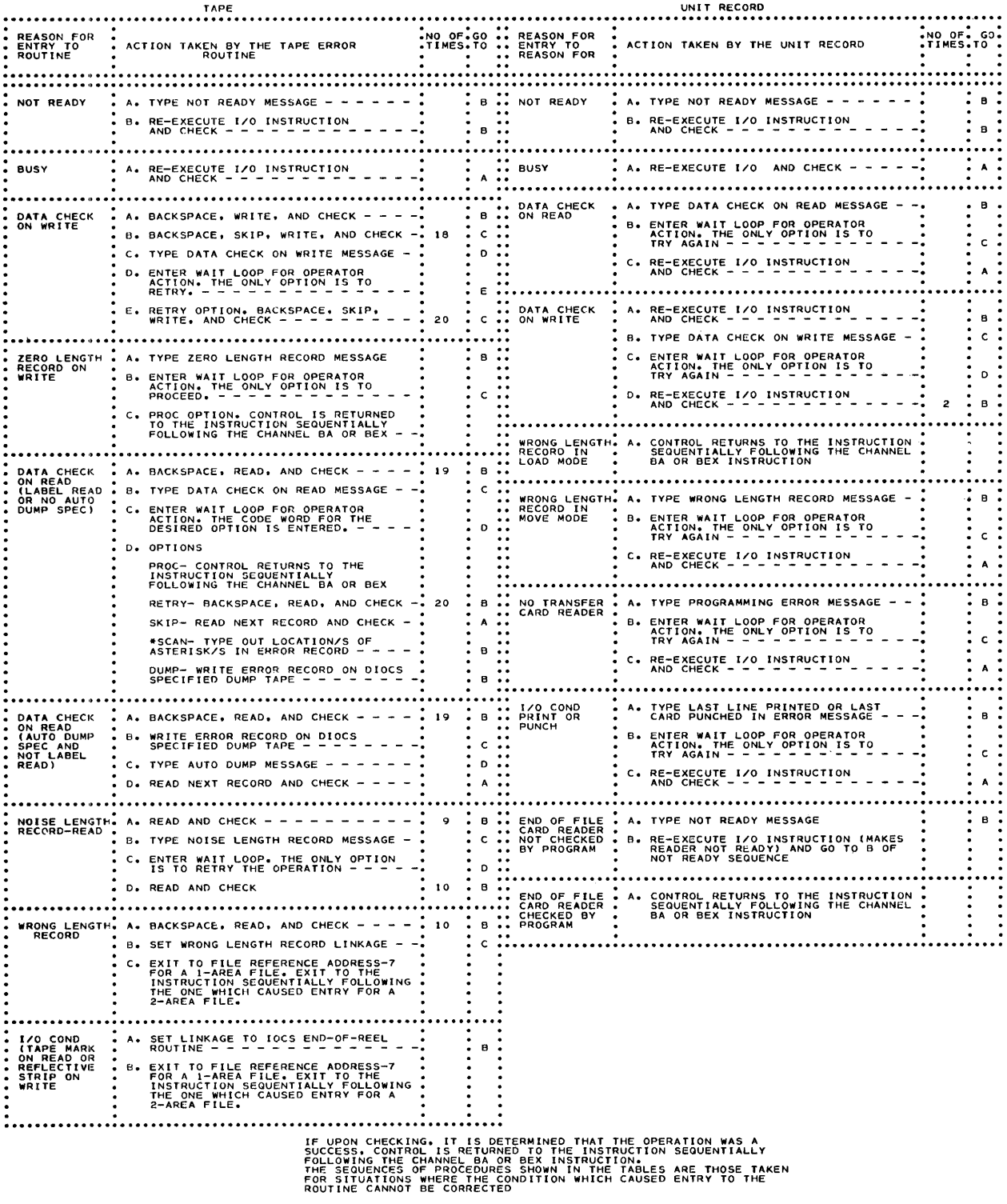
I/O CONDITION (Tape Mark on Read or Reflective Strip on Write): The routine sets up linkage between the file scheduler, from which it was entered, and the IOCS end-of-reel routine. Control returns to the instruction immediately following the Channel BA or BEX instruction for a two-area file or to the instruction at file reference address minus 7 for a one-area file.

UNIT RECORD ERROR ROUTINE

Because of the nature of unit record devices, the error routine is unable, in most instances, to take any corrective action. Therefore, the main function of the routine is to notify the operator of the type of error and to enter a wait loop for manual intervention.

The reasons for entry to the routine and the actions taken by it are shown in the unit record error routine table on Figure 1.

For a more detailed treatment, refer to Chart ED and its description.



TAPE

REASON FOR ENTRY TO ROUTINE	ACTION TAKEN BY THE TAPE ERROR ROUTINE	NO OF TIMES	GO TO
NOT READY	A. TYPE NOT READY MESSAGE		B
	B. RE-EXECUTE I/O INSTRUCTION AND CHECK		B
BUSY	A. RE-EXECUTE I/O INSTRUCTION AND CHECK		A
DATA CHECK ON WRITE	A. BACKSPACE, WRITE, AND CHECK		B
	B. BACKSPACE, SKIP, WRITE, AND CHECK	18	C
	C. TYPE DATA CHECK ON WRITE MESSAGE		D
	D. ENTER WAIT LOOP FOR OPERATOR ACTION. THE ONLY OPTION IS TO RETRY.		E
	E. RETRY OPTION. BACKSPACE, SKIP, WRITE, AND CHECK	20	C
ZERO LENGTH RECORD ON WRITE	A. TYPE ZERO LENGTH RECORD MESSAGE		B
	B. ENTER WAIT LOOP FOR OPERATOR ACTION. THE ONLY OPTION IS TO PROCEED.		C
	C. PROC OPTION. CONTROL IS RETURNED TO THE INSTRUCTION SEQUENTIALLY FOLLOWING THE CHANNEL BA OR BEX		
DATA CHECK ON READ (LABEL READ OR NO AUTO DUMP SPEC)	A. BACKSPACE, READ, AND CHECK	19	B
	B. TYPE DATA CHECK ON READ MESSAGE		C
	C. ENTER WAIT LOOP FOR OPERATOR ACTION. THE CODE WORD FOR THE DESIRED OPTION IS ENTERED.		D
	D. OPTIONS		
	PROC- CONTROL RETURNS TO THE INSTRUCTION SEQUENTIALLY FOLLOWING THE CHANNEL BA OR BEX		
	RETRY- BACKSPACE, READ, AND CHECK	20	B
	SKIP- READ NEXT RECORD AND CHECK		A
	*SCAN- TYPE OUT LOCATION/S OF ASTERISK/S IN ERROR RECORD		B
	DUMP- WRITE ERROR RECORD ON DIOCS SPECIFIED DUMP TAPE		B
DATA CHECK ON READ (AUTO DUMP SPEC AND NOT LABEL READ)	A. BACKSPACE, READ, AND CHECK	19	B
	B. WRITE ERROR RECORD ON DIOCS SPECIFIED DUMP TAPE		C
	C. TYPE AUTO DUMP MESSAGE		D
	D. READ NEXT RECORD AND CHECK		A
NOISE LENGTH RECORD-READ	A. READ AND CHECK	9	B
	B. TYPE NOISE LENGTH RECORD MESSAGE		C
	C. ENTER WAIT LOOP. THE ONLY OPTION IS TO RETRY THE OPERATION		D
	D. READ AND CHECK	10	B
WRONG LENGTH RECORD	A. BACKSPACE, READ, AND CHECK	10	B
	B. SET WRONG LENGTH RECORD LINKAGE		C
	C. EXIT TO FILE REFERENCE ADDRESS-7 FOR A 1-AREA FILE. EXIT TO THE INSTRUCTION SEQUENTIALLY FOLLOWING THE ONE WHICH CAUSED ENTRY FOR A 2-AREA FILE.		
I/O COND (TAPE MARK ON READ OR REFLECTIVE STRIP ON WRITE	A. SET LINKAGE TO IOCS END-OF-REEL ROUTINE		B
	B. EXIT TO FILE REFERENCE ADDRESS-7 FOR A 1-AREA FILE. EXIT TO THE INSTRUCTION SEQUENTIALLY FOLLOWING THE ONE WHICH CAUSED ENTRY FOR A 2-AREA FILE.		

UNIT RECORD

REASON FOR ENTRY TO REASON FOR	ACTION TAKEN BY THE UNIT RECORD	NO OF TIMES	GO TO
NOT READY	A. TYPE NOT READY MESSAGE		B
	B. RE-EXECUTE I/O INSTRUCTION AND CHECK		B
BUSY	A. RE-EXECUTE I/O AND CHECK		A
DATA CHECK ON READ	A. TYPE DATA CHECK ON READ MESSAGE		B
	B. ENTER WAIT LOOP FOR OPERATOR ACTION. THE ONLY OPTION IS TO TRY AGAIN		C
	C. RE-EXECUTE I/O INSTRUCTION AND CHECK		A
DATA CHECK ON WRITE	A. RE-EXECUTE I/O INSTRUCTION AND CHECK		B
	B. TYPE DATA CHECK ON WRITE MESSAGE		C
	C. ENTER WAIT LOOP FOR OPERATOR ACTION. THE ONLY OPTION IS TO TRY AGAIN		D
	D. RE-EXECUTE I/O INSTRUCTION AND CHECK	2	B
WRONG LENGTH RECORD IN LOAD MODE	A. CONTROL RETURNS TO THE INSTRUCTION SEQUENTIALLY FOLLOWING THE CHANNEL BA OR BEX INSTRUCTION		
WRONG LENGTH RECORD IN MOVE MODE	A. TYPE WRONG LENGTH RECORD MESSAGE		B
	B. ENTER WAIT LOOP FOR OPERATOR ACTION. THE ONLY OPTION IS TO TRY AGAIN		C
	C. RE-EXECUTE I/O INSTRUCTION AND CHECK		A
NO TRANSFER CARD READER	A. TYPE PROGRAMMING ERROR MESSAGE		B
	B. ENTER WAIT LOOP FOR OPERATOR ACTION. THE ONLY OPTION IS TO TRY AGAIN		C
	C. RE-EXECUTE I/O INSTRUCTION AND CHECK		A
I/O COND PRINT OR PUNCH	A. TYPE LAST LINE PRINTED OR LAST CARD PUNCHED IN ERROR MESSAGE		B
	B. ENTER WAIT LOOP FOR OPERATOR ACTION. THE ONLY OPTION IS TO TRY AGAIN		C
	C. RE-EXECUTE I/O INSTRUCTION AND CHECK		A
END OF FILE CARD READER NOT CHECKED BY PROGRAM	A. TYPE NOT READY MESSAGE		B
	B. RE-EXECUTE I/O INSTRUCTION (MAKES READER NOT READY) AND GO TO B OF NOT READY SEQUENCE		
END OF FILE CARD READER CHECKED BY PROGRAM	A. CONTROL RETURNS TO THE INSTRUCTION SEQUENTIALLY FOLLOWING THE CHANNEL BA OR BEX INSTRUCTION		

IF UPON CHECKING, IT IS DETERMINED THAT THE OPERATION WAS A SUCCESS, CONTROL IS RETURNED TO THE INSTRUCTION SEQUENTIALLY FOLLOWING THE CHANNEL BA OR BEX INSTRUCTION.
THE SEQUENCES OF PROCEDURES SHOWN IN THE TABLES ARE THOSE TAKEN FOR SITUATIONS WHERE THE CONDITION WHICH CAUSED ENTRY TO THE ROUTINE CANNOT BE CORRECTED

Figure 1. Tape and Unit Record Error Routine Tables

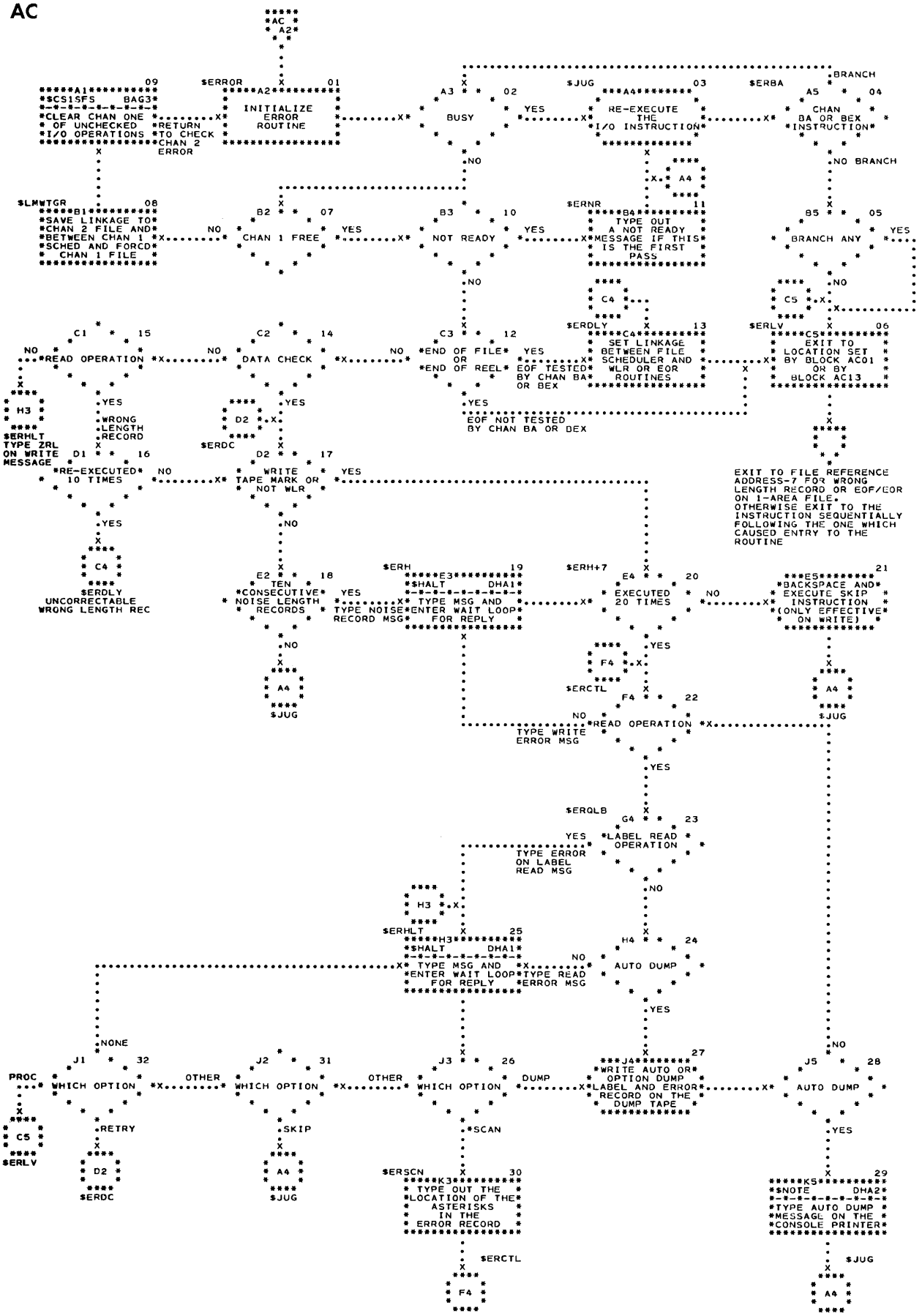


Chart AC. Tape Error Routine — Overall

The IOCS operations, treated on an over-all basis by Chart AA, Chart AB, and Figure 1, are covered on a lower level and supplemented by the following block-by-block descriptions and detailed charts. Scheduling is covered by Charts BA, BB, BC, BD, BE, and BF; record processing by Charts CA, CB, and CE; OPEN, CLOSE, FEORL, RDLIN, and end of reel by Charts DA, DB, DC, DD, DE, DF, DG, and DH; and the error routines by Charts EA, EB, EC, and ED. In these charts, much of file processing depends on the file reference table. The reader may find it helpful to refer to this table, included as Figure 7 and described in the Program Condition Analysis Aids section of the manual.

The reader is again reminded that in a label, the convention of two hyphens (--) designates the file prefix (e.g., $--FULL) either generated by IOCS or specified by the user. One hyphen (-) designates the channel prefix (e.g., $CS-SFS). The dollar sign ($) substitutes for the first four characters in the label, viz., IOCS.

Scheduling

Channel Schedulers, Including Interrupt

Block BA01, 00101: A machine interrupt, which occurs in priority alert mode when an I/O operation has been completed and an interruptable instruction is encountered, causes an automatic branch to core location 00101 and an exit from priority alert mode. At location 00101, the B-address register is stored, saving the seventh character of the interrupted instruction, and control branches to $ATTN.

Block BA02, $ATTN: The address at which the interrupt occurred is decremented by six. This adjusted address, which points to the operation code of the interrupted instruction, is stored in $INTEXT, block BA20. Control branches to $CSIENT, block BA05.

Block BA03, $ENTRY: The normal entrance to the channel scheduler from end of macro operations is made at $ENTRY. The return linkage is set in $INTEXT, block BA20.

Block BA04: If $ENTRY has been disabled, the channels are not available, and control returns directly to the I/O request without altering the mode of operation, i.e., without going through $INTEXT, which causes entry to priority alert mode. Disabling of channel operations is caused by clearing a word mark at $ENTRY+8.

Block BA05, $CS1ENT: The BOL1 determines if channel 1 is busy. If it is, control passes to $CS2ENT, block BA18. For interrupt operation, a busy condition means it cannot have been a channel 1 operation which caused the interrupt. For normal operation (start channels), a busy condition means channel 1 is already in operation.

Block BA06, $CS1PR: Channel 1 is not busy. A BOPR1 is executed. If a branch is taken, control goes to the file scheduler coding (represented by block BA10) that makes the status check for the file. The linkage to the proper scheduler was set into the BOPR1 branch address by $CS1RET, block BA17 immediately after the tape operation was started.

Block BA07, $CS1SCN: This block represents the force switch. If the switch is on, it is a forcing operation, and control goes to $CS1SF, block BA13.

Block BA08: It is not a forcing operation. The necessary branch or priority request instructions are executed to determine if there are any other interrupts on this channel, e.g., console, real time. If there is any priority request, it is served by the appropriate routine.

Block BA09, $CS1S3: This block represents the exit from the channel scheduler to the highest priority pending switch on the channel. The pending switch network is represented by block BA22.

Block BA10: The status of the I/O operation is checked. For a two-area file, the pending switch for the file is turned off. Control passes to $CS1SCN, block BA07.

Block BA11, $CS1SFS: The entrance to the force routine, $CS-SFS, sets the return address by storing the contents of the B-address register in $CS-SFX, block BA16, the force exit.

Block BA12: The force switch, $CS1SCN (block BA07) is set on, i.e., set to branch to $CS1SF, block BA13.

Block BA13, $CS1SF: A BOL1 instruction is executed to determine if the channel is in use. If it is, the channel scheduler enters (or continues) a forcing loop by branching to $CS1PR, block BA06.

Block BA14: The channel is not busy. A BOPR1 is executed. If the branch is taken, control transfers, via block BA06, to the I/O condition check at block BA10 within the file scheduler. If there is no branch, indicating that the channel entered is already clear, control passes to block BA15.

Block BA15: The channel is reset to non-forcing by setting the force switch, at $CS1SCN, block BA07, off.

Block BA16, $CS1SFX: The channel scheduler returns to the file scheduler whose address was set by $CS1SFS, block BA11.

Block BA17, $CS1RET: At $CS1RET, the branch address of $CS1PR, block BA06, is set to the file scheduler coding which will make the I/O status check for the file for which a tape operation has just been started.

Block BA18, $CS2ENT: The BOL2 determines if channel 2 is busy. If it is, control branches to $INTEXT, block BA20. If the channel is not busy, control passes to block BA21. For interrupt operation, a busy condition means it cannot have a been a channel 2 operation which caused the interrupt. For normal operation (start channels), a busy condition means channel 2 is already in operation.

Block BA19, $CS2RET: At $CS2RET, the branch address of $CS2PR is set to the file scheduler coding which will make the I/O status check for the file for which a tape operation has just been started.

Block BA20, $INTEXT: At $INTEXT is the BEPA instruction which causes entry into the priority alert mode. The three possible types of exits from this block are:

1. The interrupt exit (set at blocks BA01 and BA02) returns the program to the point at which the interrupt occurred.
2. The normal exit (set at block BA03) is a return to the file scheduler as set by $ENTRY.
3. The bootstrap force exit (set by block BC05 or BD07) returns control to the file scheduler via the force routine which is entered at block BA12 and exited at block BA16.

Block BA21: With the exception of the entrance and exit, the logic of the coding is the same for channel 2 as for channel 1. In most instances, the coding itself, except for the beginning of the tags ($CS2 as against $CS1 for channel 1), is also the same.

Block BA22: The pending switch network is entered at the highest-priority pending switch on the channel. If no pending switch is on, control passes down the network along the pending switch off-branches and exits to $CS2ENT for channel 1 or to $INTEXT for channel 2. If a pending switch is on, control takes the on-branch of that switch (shown as block BC07 or BD09) to start an I/O operation.

One-Area Input File Scheduler

Block BB01, $--EMTY: Entry from a GET macro is made at $--EMTY. The return to the macro is set in $--EXIT, block BB10. Control passes to the channel scheduler (represented by block BB03) by a BXPA instruction to cause exit from the priority alert mode.

Block BB03: The appropriate channel is cleared.

Block BB04: $--IOA: The read instruction is executed.

Block BB05: The I/O status check for the channel is executed. The test is a BA if a wrong-length record check is being performed. It is a BEX if wrong-length records are not being checked for and/or if the DTF specified variable-length records. If all indicators are off, i.e., the read was valid, control passes to block BB06; if any indicator is on, control passes to the error routine represented by block BB11.

Block BB06: This coding is included only if blocked records were specified in the DTF. The area or record limits are reset. If wrong-length records and/or checkpoint records are specified for variable-length blocked records, a check is made for these records.

Block BB07, $--TRIG: The BXPA instruction at $--TRIG is a branch to $ENTRY, the entrance to the channel scheduler at block BA03. The pending I/O operation of highest priority on each channel is executed after checking the last one on the channel, if any. Priority alert mode is re-entered.

Block BB08. The block count is incremented by +1.

Block BB09, $--EXIT: Control returns to the main line program address set up at $--EMTY, block BB01.

Block BB11: The tape error routine, entered at block EA01, attempts to correct any error resulting from the tape read operation. If possible, corrective action is taken and control returns to block BB06. If an end-of-reel condition is encountered, control branches to the end-of-reel routine at block DD01; if a wrong length record is found, control branches to block BB12.

Block BB12: If checkpoint records are specified, a test is made to determine if a checkpoint record has been read. Otherwise, control goes immediately to the wrong-length record routine supplied by the user. If it is determined that this is a checkpoint record, control passes to block BB13.

Block BB13: The checkpoint record is bypassed by executing another read instruction. Control returns to block BB04 to read the next record.

Block BB14: The user's wrong length record routine determines whether to accept or reject a wrong length record. If the record is accepted, control returns to block BB06 to process the record. If the record is rejected, control returns to block BB04 to get the next record.

Block BB15: (Only for Form 4 records.) The block character count is compared to the number of characters read to determine if a wrong-length record has been read. If the record read is not as specified, control passes to the Form 4 WLR sequence at block BB16.

Block BB16, $--WLR: This is the Form 4 WLR sequence. The record in question is backspaced to enable

a re-read. If a successful read is not completed after nine tries, control branches to the user's wrong-length record routine.

One-Area Output File Scheduler

Block BB21, $--FULL: Entry from a PUT macro is made at $--FULL. The return to the macro is set in $--EXIT.

Block BB22: The block count is incremented by +1. If the file scheduler is for variable-length, blocked records, a group mark/word mark is set one location beyond the last data character position, and the block character count is placed in the first four positions of the record. Control passes to the channel scheduler (represented by block BB24) by a BXPA instruction to cause exit from the priority alert mode.

Block BB24: The appropriate channel is cleared.

Block BB25, $--IOA: The write instruction is executed.

Block BB26: For a write operation, the I/O channel status check is always a BA. If a branch occurs, control transfers to block BB31.

Block BB27: The file scheduler controls are reinitialized with the area limits for a fixed-length record or with the record-size limits for a variable-length record.

Block BB28, $--TRIG: The BXPA instruction at $--TRIG is usually a branch to $ENTRY, the entrance to the channel scheduler (represented by block BB29 on this chart). However, it can also be used as a pivot by those IOCS routines concerned with the file schedulers, e.g., end-of-reel and error routines.

Block BB29: The channel scheduler is entered at block BA03, $ENTRY. The pending I/O operation of highest priority is executed after checking the last one on the channel, if any. Priority alert mode is re-entered.

Block BB30, $--EXIT: Control returns to the mainline program location set by $--FULL, block BB21, or to the close routine as set by $--PADS, block BF01 or BF11.

Block BB31: The tape error routine, entered at block EA01, attempts to correct any error resulting from the tape write operation. If possible, corrective action is taken and control returns to block BB27. If there is an end-of-reel condition, control passes to the routine beginning at block DD01.

Two-Area Input File Scheduler

Block BC01, $--EMTY: Entry from a GET macro is made at $--EMTY. The return to the macro is set in $--EXIT, block BC24.

Block BC02, $--WTG: The pending switch, block BC07, for the file is tested. If variable-length, blocked records were specified in the DTF and wrong-length records are being checked, the wrong-length record count is set to zero prior to testing the pending switch. If the pending switch is off, the other record area is available for processing and a branch is taken to $--PA, block BC12. If the pending switch is on, the other area is not available and a forced read operation for this file must be initiated to make the area available. A BXPA, to leave the priority alert mode, is taken to the channel scheduler force routine (represented by block BC04).

Block BC04: The channel is set to forcing and is cleared.

Block BC05: The branch address of $INTEXT, block BA20, is set with the bootstrap force entry address; it is set to go to block BA12. This is to insure the completion of the I/O operation to be started.

Block BC06: A second test of the pending switch is made. For the first time through, the switch is off if the I/O operation cleared at block BC04 happened to be for this file. Otherwise, it is on and the forcing operation is continued. For the second time through (bootstrap return) after the forcing operation is completed, it is always off.

Block BC07, $-0000N: When the pending switch is NOP, it is on. When the pending switch is BRANCH, it is off. The entrance to the pending switch is either from the file scheduler itself, or from the greater priority pending switch (N−1) immediately prior to this one. If there is no pending switch of greater priority on this channel, the entrance is from the channel scheduler. If the pending switch is on, control falls through to the read operation of the file scheduler. If the pending switch is off, it is a branch to the next-lower priority pending switch (N+1) or if there are no lower priority pending switches on the channel, the branch is either to $CS2ENT or to $INTEXT, for channel 1 and channel 2, respectively.

Block BC08: A test is made to determine which area is to be used.

Blocks BC09, $--IOA, and BC10, $--IOB: These two blocks represent the actual read operations for this file. Block BC09 reads into area A; block BC10 reads into area B.

Block BC11: A test is made to determine if the read operation has been successfully started. If the operation has been started, control branches to $CS-RET in the channel scheduler where the return address to the I/O status check is set in $CS-PR, block BA06. If the read operation has not been started, the I/O status check is made immediately at block BC29.

Block BC12, $--PA: A test to determine which area was used last is made by testing for a word mark on the read operation. If a word mark is present, area B is indicated, and control passes to $--SA, block BC15.

Block BC13: Area A was read into last; therefore the switches are changed so that area B will be read into next.

Block BC14: The file scheduler controls are reinitialized with the area or record limits depending upon the use of fixed- or variable-length records, respectively. The address of the first record character is placed into $--SAVE. The ending address, $--ENDD, is set with the address of the last character in the input record. Control passes to block BC17.

Block BC15: The read operation is changed to read into area A next.

Block BC16: This is identical with block BC14, except that the source of information is the B area limits rather than the A area limits.

Block BC17: The pending switch is set on to indicate that one of the areas has been used and to request the refilling of that area.

Block BC18: This block is included only when the SOURCE DTF specifies one of the following conditions: (1) fixed-length, blocked records using index registers and checkpoints, or (2) variable-length, blocked records and checkpoints. If included, this block tests for a checkpoint record. If such a record is found, control passes to $--BYP, block BC19; otherwise, it goes to block BC20.

Block BC19, $--BYP: The checkpoint record sensed at block BC18 is bypassed.

Block BC20, $--TRIG: The BXPA instruction at $--TRIG is usually a branch to $ENTRY, the entrance to the channel scheduler (represented by block BC21 on this chart). It may also be set, for certain exceptional conditions, as a branch to the routines concerned with processing those conditions. The exceptionals conditions and the associated processing are:

End-of-Reel: After the sensing of a tape mark, $--TRIG is set by the error routine to branch to end-of-reel procedures at $EORU, block DD01.

Prime Operation: The execution of open procedures (in an OPEH operation, or while opening a new reel in a FEORL or end-of-reel operation) causes $--TRIG to be set to branch to the linkage, block BC25, to the prime routine.

WLR Operation for Fixed-Length Records: After ten additional unsuccessful attempts to read the record in question, the error routine sets $--TRIG to branch to the linkage, block BC25, to the user's WLR routine.

Block BC21: This block is included only for variable-length, blocked records. If a wrong-length record check is specified, the check is made at this point. A branch to the Form 4 WLR sequence, block BC35, is made if a wrong-length record is found.

Block BC22: If the channel is free, an I/O operation is started either because of the pending request for this file or for higher-priority file. The priority alert mode is re-entered.

Block BC23: The block count is incremented by +1.

Block BC24, $--EXIT: The return is to the GET macro as set by block BC01.

Block BC25: The purpose of the coding in this block is to reset $--TRIG to the normal exit, i.e., a BXPA to $ENTRY. The two possible exits from this block are (1) to the wrong-length record routine, block BC27, or (2) to the IOCS prime routine, block BC26.

Block BC26: To ensure correct usage as a two-area file, a priming of this file is forced. Control passes to block BC02.

Block BC27: A test is made to determine if a checkpoint header record was read. If so, the checkpoint record itself is read. Control then returns to $--WTG, block BC02, to read a data record into the file.

Block BC28: The user's wrong-length record routine determines if the record should be accepted or rejected. If the record is rejected, control returns to $--WTG, block BC02. If the record is accepted, control returns to $--TRIG, block BC20.

Block BC29: The I/O channel status check for this file is a BA instruction if wrong-length records are being checked or a BEX instruction if there is no wrong-length record check. If it is a BEX instruction, the I/O interlock has not yet been turned off, unless a branch is taken on the particular indicator. Control passes to block BC30 if an error is sensed; otherwise, it goes to block BC31.

Block BC30: The tape error routine is entered at block EA01. An attempt is made to correct any error or $--TRIG is set to indicate and end-of-file or wrong-length record condition.

Block BC31: A branch any to self plus one to prevent the machine I/O interlock is executed if wrong-length records were not checked for.

Block BC32: If there are variable-length, blocked records and wrong-length records are to be checked for, the E or F register is stored to enable the wrong-length check.

Block BC33: The pending switch is turned off.

Block BC34: The channel scheduler is entered at BA07. Any pending I/O on this channel is started or the channel is cleared and control returns to the file scheduler. There are two possible returns. One is a return to the main program through $INTEXT, block BA20, if this file scheduler was entered from an interrupt. The other returns is to the file scheduler at block BC05 via $INTEXT and the bootstrap force routine.

Block BC35, $--WLR: This is the Form 4 WLR sequence. The record in question is backspaced to enable a re-read. If a successful read is not completed after nine tries, control branches to the user's wrong-length record routine, block BC28.

Two-Area Output File Scheduler

Block BD01, $--FULL: Entry from a PUT macro is made at $--FULL when the current record area has been filled with logical records. The return to the macro is set in $--EXIT, block BD19.

Block BD02: The block count, accumulated in $--TBC, is increased by 1.

Block BD03: This block is pertinent only for a variable-length, blocked record file. A group mark/word mark is placed at one location beyond that of the last character of the blocked record. The block character count is placed in the first four positions of the record to enable the correct handling of this record.

Block BD04, $--WTG: The pending switch, block BD09, is tested. If the pending switch is off, the other record area is available for processing (filling) and a branch is taken to $--PA, block BD10. If the pending switch is on, the other area is not available. A forced write operation for this file must be initiated to make the area available. A BXPA, to leave the priority alert mode, is taken to the channel force routine (represented by block BD06).

Block BD06: The channel is set to forcing and is cleared.

Block BD07: The branch address of $INTEXT, block BA20, is set with the bootstrap force entry address, i.e., set to go to block BA12. This is to insure the completion of the I/O operation to be started.

Block BD08: A second test of the pending switch is made. For the first time through, it is off if the I/O operation cleared at block BD06 happened to be for this file. Otherwise, it is on and the forcing operation is continued. For the second time through (bootstrap return) after the forcing operation is completed, it is always off.

Block BD09, $-0000N: When the pending switch is at NOP, it is considered to be on. When the pending switch is a branch instruction, it is considered off. The entrance to the pending switch is either from the file scheduler itself, or from the greater priority pending switch (N−1) immediately prior to this one. If there is no pending switch of greater priority on this channel, the entrance is from the channel scheduler. If the pending switch is on, control falls through to the write operation of the file scheduler. If the pending switch is off, it is a branch to the next-lower priority pending switch (N+1) or if there are no lower priority pending switches on the channel, the branch is either to $CS2ENT or to $INTEXT, for channel 1 and channel 2 respectively.

Block BD10, $--PA: The area last used is determined by testing for a word mark. If area B was last used, control passes to $--SA, block BD13; otherwise, it goes to block BD11.

Block BD11: Area A was written from last; therefore, the switches are changed so that area B will be written from next.

Block BD12: The file scheduler controls are reinitialized with the area or record limits depending upon the use of fixed- or variable-length records. The address of the first record character is placed into $--SAVE. The ending address, $--ENDD, is set with the address of the last character in the record. Control passes to block BD15.

Blocks BD13, $--SA, and BD14: These blocks are essentially the same as blocks BD11 and BD12 except that the change is from utilization of area B to area A and the limits set are those for area B.

Block BD15: The pending switch is set on so as to indicate that one of the areas has been used and to request the writing of it.

Block BD16, $--TRIG: The BXPA instruction at $--TRIG is usually a branch to $ENTRY, block BD17. It may also be set, for certain exceptional conditions, as a branch to those routines concerned with processing those conditions. The exceptional conditions and the associated processing are discussed below:

End of Reel: After a reflective spot is sensed, $--TRIG is set by the error routine as a branch to $EORU, block DD01.

CLOSE and FEORL Operations: $--TRIG is set by block DE11 in the close procedures to return to those procedures after writing the final block of records (if any).

Block BD17: If the channel is free, an I/O operation is started either because of the pending request for this file or for a higher-priority file. The priority alert mode is re-entered.

Block BD18: This block is applicable only for variable-length, blocked records. The block character count is set to zero.

Block BD19, $--EXIT: Control returns to the PUT macro as set by block BD01.

Block BD21: The area from which data are to be written is determined. For area A, control passes to block BD22; for area B, it goes to block BD23.

Block BD22, $--IOA, and Block BD23, $--IOB: These two blocks represent the actual write operations for the file. Block BD22 writes from area A; block BD23 writes from area B.

Block BD24: A test is made to determine if the write operation has been successfully started. If the operation has been started, control branches to $CS-RET in the channel scheduler where the return address to the I/O status check is set in $CS-PR, block BA06. If the read operation has not been started, the I/O status check is made immediately, block BD25.

Block BD25: For a write operation, the I/O channel status check is always a BA. If a branch occurs, control transfers to block BD26. If no branch occurs, control passes to block BD27.

Block BD26: This block represents the error routine. If the error is correctable, it is rectified and control is returned to block BD27. An attempt is made to correct any error or $--TRIG is set to indicate an end-of-file or wrong-length record condition.

Block BD27: The pending switch is set off.

Block BD28: This block represents the channel scheduler, entered at BA07. Any pending I/O on this channel is started or the channel is cleared and control returns to the file scheduler. There are two possible exits. One is a return to the main program via $INTEXT, block BA20, if this file scheduler was entered from an interrupt. If this file scheduler was in a forcing position, the other return is to the file scheduler at block BD07 via $INTEXT and the bootstrap force routine.

Tape File Initialization Sequence

The tape file initialization routines are described in a logical rather than a coding fashion and are based on a pseudo decision table containing in its condition stub entries from the DTF for the file. The number of areas used, the record format, and, for input files, whether wrong-length records are checked, define the coding blocks required for initialization. All possible coding blocks are shown in an in-line, logical sequence, but all blocks will not be included for any given file.

Blocks BE06 through BE09 and blocks BE17 through BE19 will not be generated for a file unless the DIOCS contains a CHANCHANGE entry; this condition overrides any specifications for block inclusion which are derived from the decision table.

INPUT FILE SCHEDULER INITIALIZATION SEQUENCE

Block BE01, $--INIT, and Block BE02: The operation code and the X-control field (from $--ACT+6, the base tape identification for the file) are moved into $--IOA+6 (and $--IOB+6), the area A (area B) read instruction(s) for the file.

Block BE03: (Included only for blocked records.) $--SAVE is loaded with the address of the last location of area A.

Block BE04: (Included only for a one-area, variable-length, blocked file.) $--ENDD is set to zero. This field will contain the block character count for each tape record.

Block BE05: (Included only for two-area files.) $--TRIG, block BC20, is set to enter the priming routine, i.e., to branch to $--PRIM.

Block BE06: The branch address of $--SFX, the file scheduler linkage to the channel scheduler force routine, is reset to refer to the proper channel.

Block BE07: (Included only for two-area files) The I/O status check, block BC29, is reset. It may now require a different channel operation code. The file scheduler return to the channel scheduler ($CS-RET, block BA17 or BA19), is reset to the proper channel.

Block BE08: (Included only for one-area, blocked files): The I/O status check is reset as it may now require a different channel operation code.

Block BE09 (Included only when the Form 4 WLR routine is used): The d-modifier of the store E- (F-) address register instruction is reset so as to store the proper register.

Block BE10: Control returns to the common initialization sequence beginning at $ENTAB, block DB11.

OUTPUT FILE INITIALIZATION SEQUENCE

Block BE11, $--INIT, and Block BE12: The operation code and the X-control field (from $--ACT+6, the base tape identification for the file) are moved into $--IOA+6 (and $--IOB+6), the area A (area B) write instruction(s) for the file.

Block BE13: $--SAVE is loaded with the address of the first character of area A.

Block BE14: (Included only for a variable, blocked file.) $--SAVE is loaded with the address of the block character count. The record length is set to plus zero.

Block BE15: (Included only for blocked, fixed-length records.) $--ENDD is loaded with the address of the last character of area A.

Block BE16: (Included only for two-area files.) $--TRIG, block BD16, is set to branch to $ENTRY, block BA03.

Block BE17: The branch address of $--SFX, the file scheduler linkage to the channel scheduler force routine, is reset to refer to the proper channel.

Block BE18: (Included only for two-area files.) The I/O status check, block BD25, is reset. It may now require a different channel operation code. The file scheduler return to the channel scheduler ($CS-RET at block BA17 or BA19) is also reset to the proper channel.

Block BE19: (Included only for one-area, blocked files.) The I/O status check is reset in a fashion similar to that of block BE07.

Block BE20: Control returns to the common initialization sequence beginning at $ENTAB, block DB11.

Padding Routines

PADDING ROUTINE FOR A FILE USING AN INDEX REGISTER

Block BF01, $--PADS: Entry is from the RELSE macro or from the close procedures; $--EXIT (the exit from the associated file scheduler) is set to return to the proper routine.

Block BF02: A test is made to determine whether the output block requires padding. If no padding is required (i.e., the area index register points to the end of the area), control returns to the proper routine via the $--EXIT address set upon entering. If the file needs padding, control passes to $--PLB2, block BF03.

Block BF03, $--PLB2: If the index register is pointing at a record mark (end of record), control branches to block BF06.

Block BF04: The area end address is compared to the address in the index register to determine if additional padding is required. If no more padding is needed, the record is ready to be written, and control branches to block BF07. If more padding is needed, control passes to block BF05.

Block BF05: The padding character is moved to the address specified by the index register.

Block BF06, $--PTRC or $--PLB1: The index register is increased by 1. Control branches to block BF03.

Block BF07: This block represents the file scheduler entered at $--FULL+7, blocks BB22 or BD02. After the padded block is written, control returns to the proper routine as set by block BF01.

PADDING ROUTINE FOR A FILE NOT USING AN INDEX REGISTER

Block BF11, $--PADS: Entry is from the RELSE macro or from the close procedures; $--EXIT (the exit from the associated file scheduler) is loaded with the return address.

Block BF12: This is a test to determine if the output block needs padding. This is accomplished by comparing the address of the last character put into the output area to the address which defines the end of the area. If no padding is needed, control returns to the proper routine via $--EXIT. If padding is required, control passes to block BF13.

Block BF13: A word mark is moved to the high-order position of $--PSVE. This sets up an area where the contents of X15 can be saved.

Block BF14: The contents of X15 are stored in $--PSVE. Index register 15 is then loaded with the contents of $--SAVE.

Block BF15, $--PLB2: If X15 is pointing at a record mark (end of record), control branches to block BF18.

Block BF16: The contents of X15 are compared to the final address of the area to determine if the file is ready to be written. If it is, control branches to block BF19.

Block BF17: The padding character is moved to the address specified by index register 15.

Block BF18, $--PTRC or $--PLB1: The contents of X15 are increased by 1. Control returns to block BF15.

Block BF19: The contents of $--PSVE are restored to X15.

Block BF20: This block represents the file scheduler entered at $--FULL+7, block BB22 or BD02. After the padded block is written, control returns to the proper routine as set by block BF11.

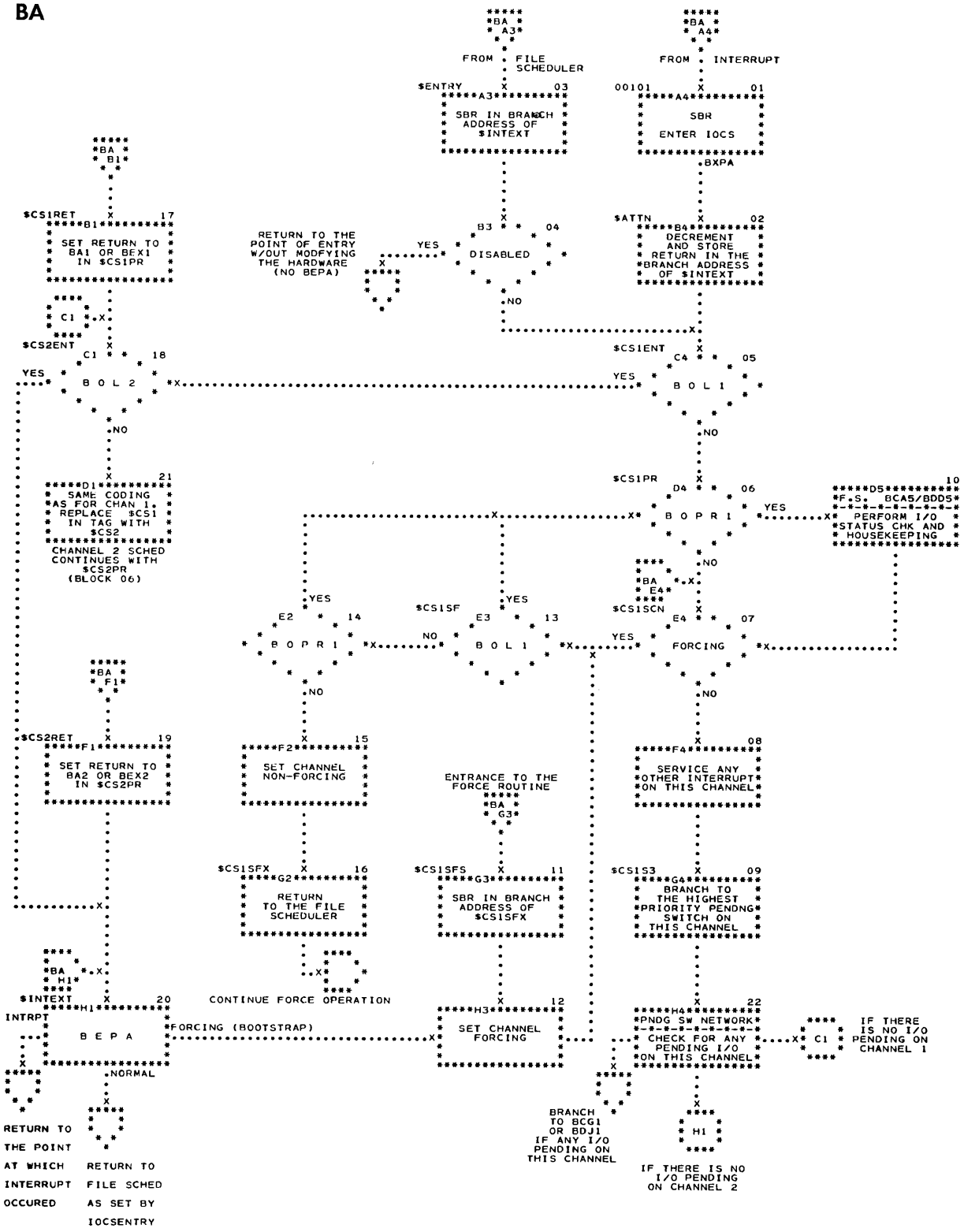


Chart BA. Channel Schedulers, Including Interrupt

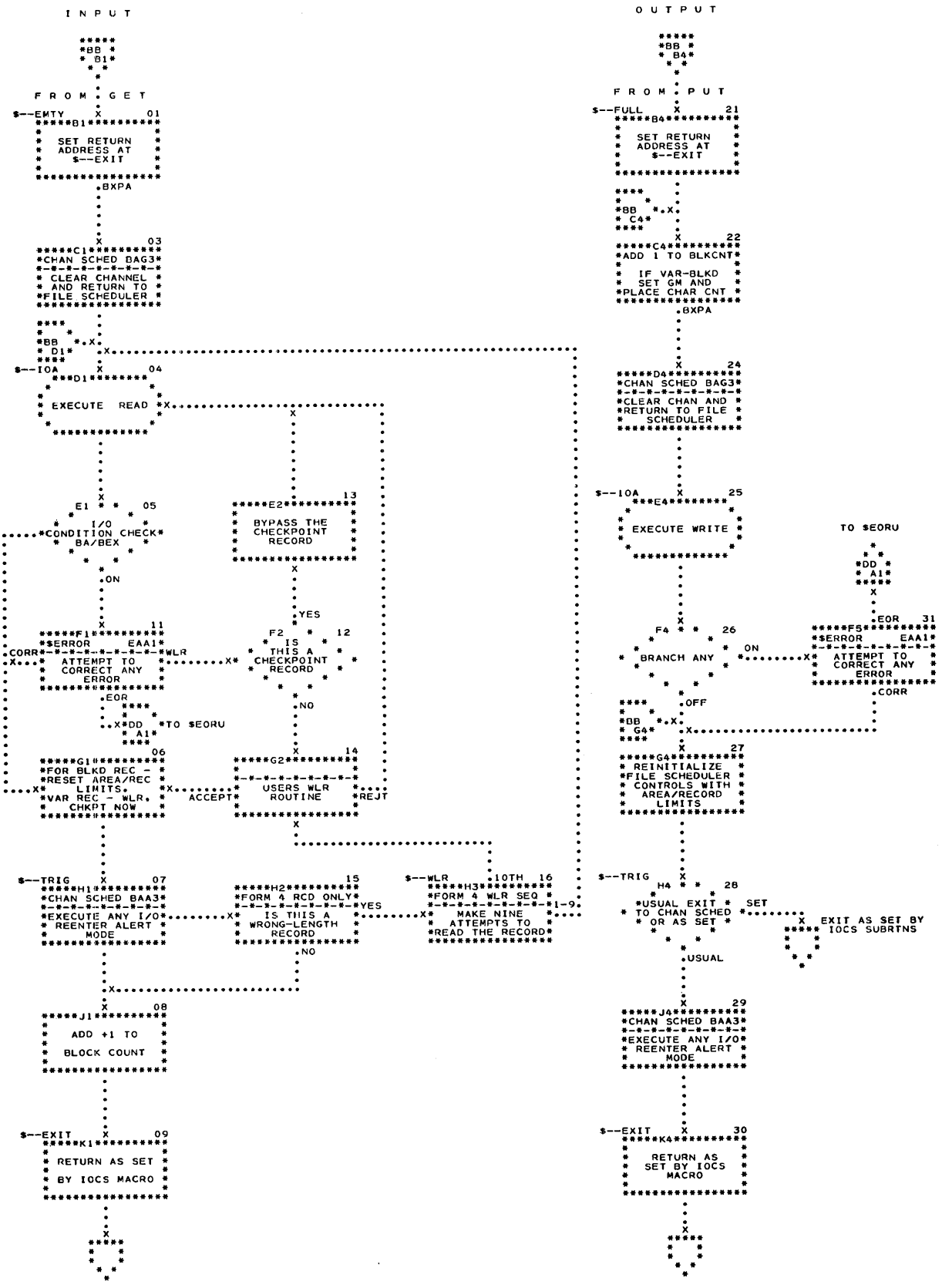
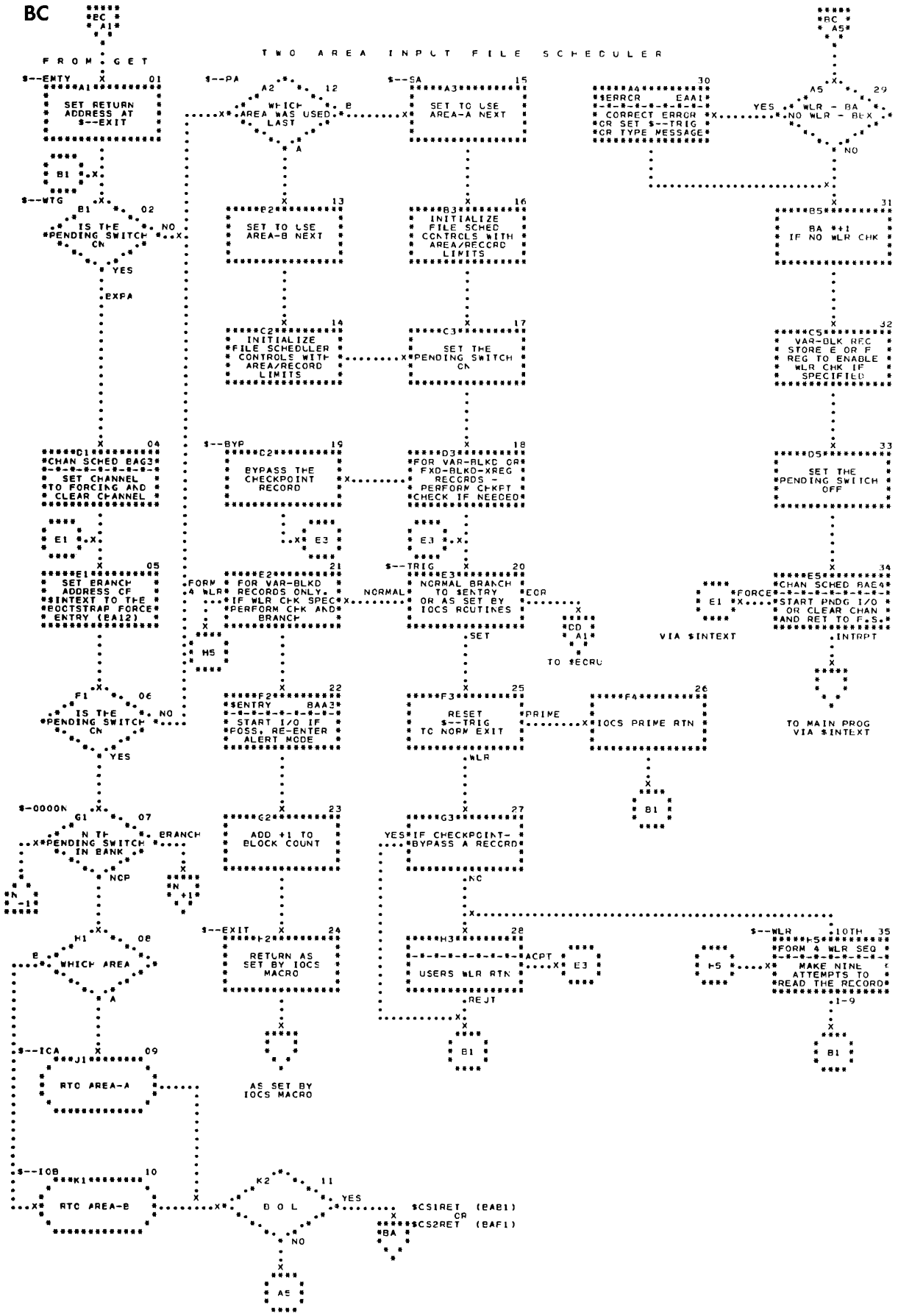


Chart BB. One-Area File Schedulers — Input and Output



- Chart BC. Two-Area Input File Scheduler

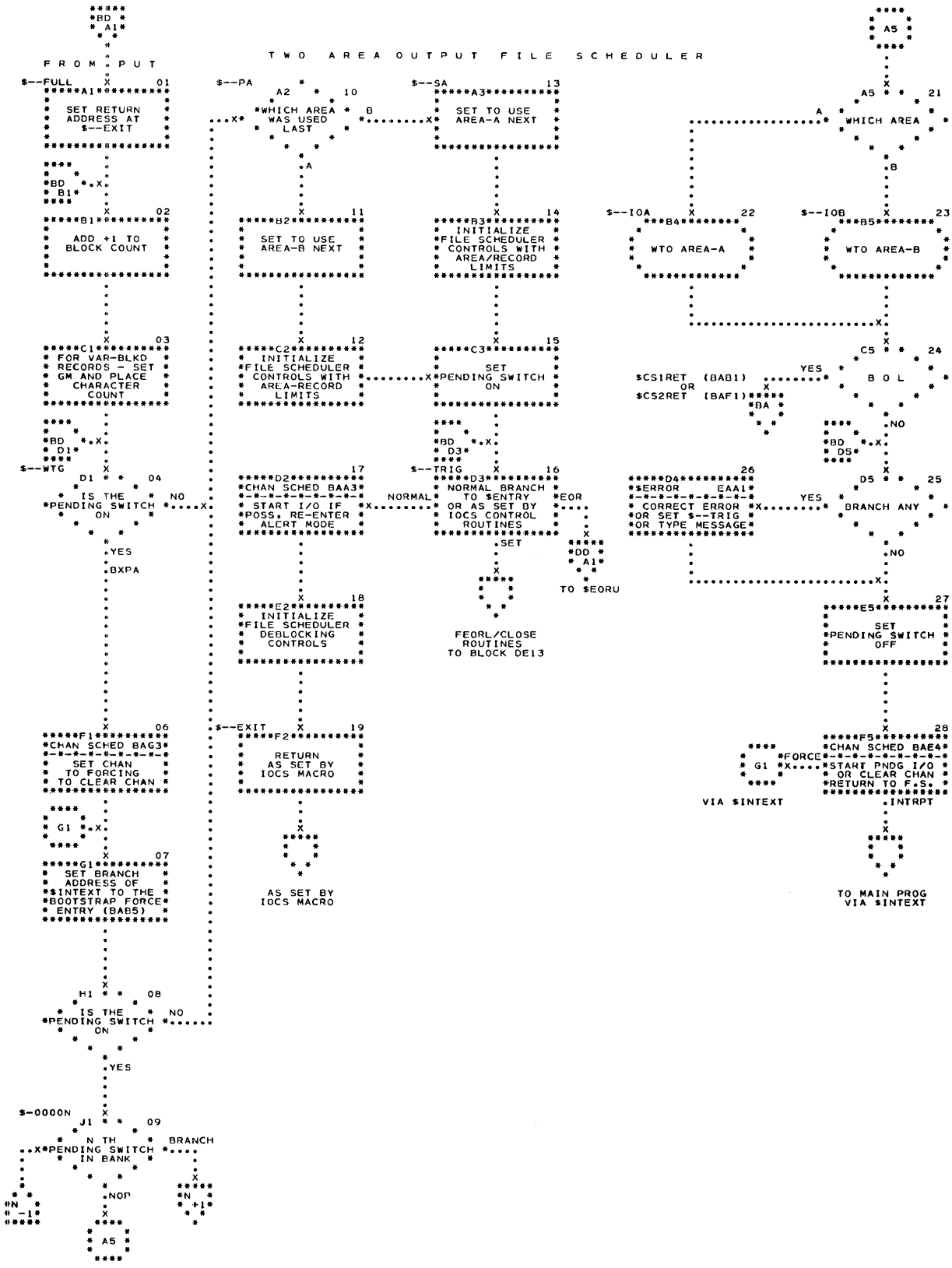


Chart BD. Two-Area Output File Scheduler

INPUT FILE SCHEDULER

NO OF AREAS	BLKNG USED	RECRD FORM	WLR CHKD	CODING BLOCKS GENERATED
1	NO	N/A	NO	01,06,08,10
1	NO	N/A	YES	01,06,08,09,10
1	YES	FXD	NO	01,03,06,08,10
1	YES	FXD	YES	01,03,06,08,09,10
1	YES	VAR	NO	01,03,04,06,08,10
1	YES	VAR	YES	01,03,04,06,08,09,10
2	NO	N/A	NO	01,02,05,06,07,10
2	NO	N/A	YES	01,02,05,06,07,09,10
2	YES	N/A	NO	01,02,03,04,05,06,07,10
2	YES	N/A	YES	01,02,03,04,05,06,07,09,10

OUTPUT FILE SCHEDULER

NO. OF AREAS	BLKING USED	RECORD FORMAT	CODING BLOCKS GENERATED
1	NO	N/A	11,17,19,20
1	YES	FXD	11,13,17,19,20
1	YES	VAR	11,14,17,19,20
2	NO	N/A	11,12,13,16,17,18,20
2	YES	FXD	11,12,13,15,16,17,18,20
2	YES	VAR	11,12,14,16,17,18,20

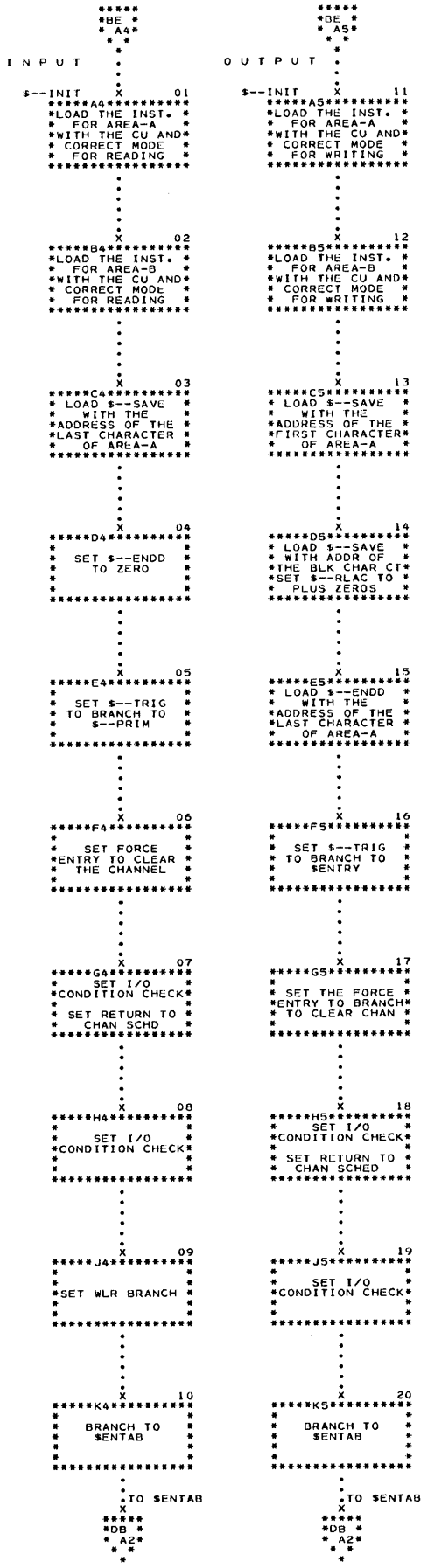


Chart BE. Tape File Initialization Routines

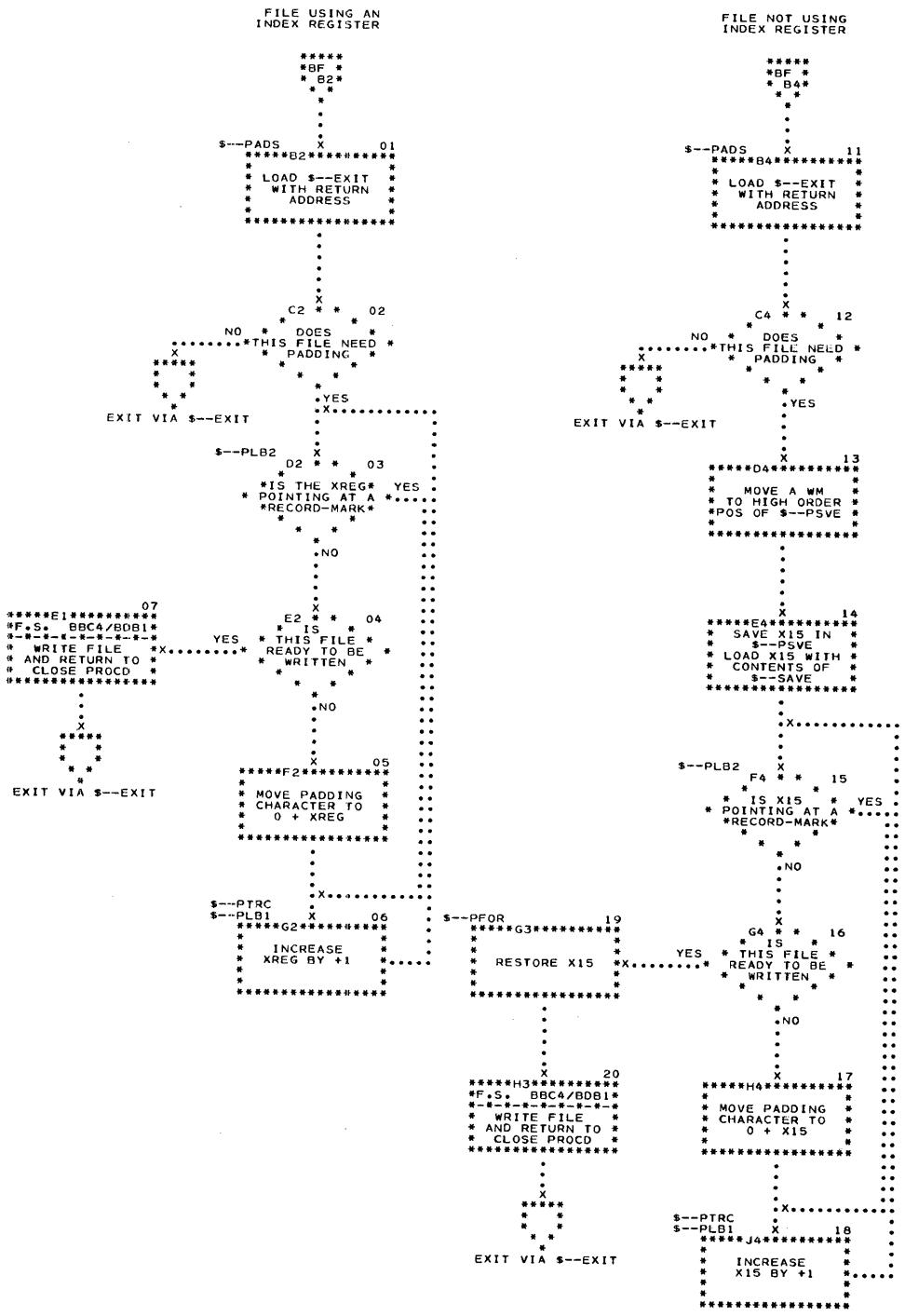


Chart BF. Padding Routines

Record Processing and Little Macros

The record processing macros for tape files with specified DTF's are GET, PUT, and RELSE shown on Chart CA. The GET macro, sometimes referred to as a deblocking routine, makes the next logical record in a block available for processing; a PUT adds the next logical record to a block. The many formats for GET and PUT are covered. The record processing macros for unit record files with specified DTF's are shown on Chart CB. The little macros RTAPE, WTAPE, CONSL, RTLBL, WTLBL, STACK, SKIP, IOBSP, IORWD, IORWU, IOWTM, IOSYS, and PSTAC are shown on Chart CC.

PUT, GET, and RELSE Macros

The PUT, GET, and RELSE macros are charted logically, independent of generated coding. In other words, the blocks do not represent actual coding but represent the logic behind the coding. The description indicates some of the possibilities that can be coded. The PUT and GET macros, in particular, vary from file to file, dependent upon the source DTF specifications and format of the PUT or GET being used. For example, if a work area were PUT to a file, the coding is different than if a file were simply PUT or if a different file were PUT to that file. The coding also differs if the file is blocked or unblocked, variable or fixed, and uses one or two areas. Similarly, for the GET macro the coding is different depending upon format and the usage of index words, blocked records, or variable records. The RELSE macro usually causes only a few instructions to be generated. Essentially, it is the replacement of the current logical record address with the ending (input) or the writing of the record (output).

PUT MACRO

Block CA01: This is a logical in-line connection to the user's coding.

Block CA02: The initialization for a move may be the moving of the address from $--SAVE to an index register, or to the actual move instruction. It is not generated if the move does not need initialization, e.g., if both files use index words and file 1 is being PUT to file 2.

Block CA03: The actual movement of the record may not be a physical movement of the data but a movement of addresses for the next logical record.

Block CA04: In the updating for the next PUT macro, the addresses that will be used for the file being PUT are increased to reflect the correct location. In other words, $--SAVE is increased by the length of the record.

Block CA05: If hash totals have been specified, the contents of the field designed for hash total is added to $--THT.

Block CA06: If record counts have been specified, a +1 is added to $--TRC.

Block CA07: $--SAVE is tested to determine if the current I/O area is available. If it is, control returns to the user's program. Otherwise, the area is full and the file scheduler is entered at $--FULL (represented by block CA08).

Block CA08: (Two-area.) After the other area is made available to the macro (by a forced write operation if necessary), the file is set pending to record the need to write the full area. *(One-area.)* The area is written to make it available again.

Block CA09: This represents the logical in-line connection with the user's coding.

GET MACRO

Block CA11: This represents the logical in-line connection with the user's coding.

Block CA12: A test is made to determine if the current I/O area is available. If it is, control branches to block CA14. Otherwise, the area is empty, and the file scheduler is entered at $--EMTY (represented by block CA13).

Block CA13: (Two-area.) After the other area is made available to the macro (by a forced read operation if necessary), the file is set pending to record the need to read into the empty area. *(One-area.)* The area is read into to make it available again.

Block CA14: The initialization for a move may be the moving of the address from $--SAVE to an index register, or to the actual move instruction.

Block CA15: The actual movement of the record may not be a physical movement of the data but a movement of addresses for the next logical record.

Block CA16: In the updating for the next GET macro, the addresses that will be used for the file are increased to reflect the correct location. In other words, $--SAVE is increased by the length of the record.

Block CA17: Hash totals, if they have been specified, are accumulated in $--THT.

Block CA18: If record counts are specified, a +1 is added to $--TRC.

Block CA19: This is the in-line connection with the user's coding.

RELSE MACRO, INPUT

Block CA21: This represents the in-line connection with the user's program.

Block CA22: The address of the logical record being operated upon is replaced by the address of the last character in the area.

Block CA23: This is the logical connection to the user's program.

RELSE MACRO, OUTPUT

Block CA24: This represents the in-line connection with the user's program.

Block CA25: Control transfers to the padding routine for this file. If the block requires padding, it is padded before being written.

Block CA26: This represents the in-line connection with the user's program.

Unit Record GET, PUT, and Close Operations

The function of unit record file schedulers, when used with GET/PUT macros, is to execute and check the I/O operation requested by the macro, keep an accumulative count of the number of records, and, if applicable, provide linkage to the user's end of file routine.

An additional function of the punch file scheduler, when entered during a close macro, is to punch a blank card so that all the punched data cards are in the stacker when the punch file is closed.

A detailed treatment of the file schedulers and the GET/PUT macros follows.

GET CARD MACRO AND SCHEDULER

Block CB01: If the GET macro specifies an end-of-file address for the card reader file, the specified EOFADDR address is placed in the I-address of the file scheduler's end-of-file test instruction (block CB07). Note that this replaces the DTF EOFADDR if one had been specified. (Refer to the description of block CB07.)

Block CB02, $--EMTY: This is the labeled entry to the file scheduler. The contents of the B-address register are stored in the scheduler's exit, $--EXIT (block CB12).

Block CB03, $--SFX: A branch and exit priority alert mode to $CS-SFS, the force entry to the channel scheduler, is executed. The channel for which the GET macro was issued is forced clear of all unchecked I/O operations. This is done to prevent an I/O interlock when the card read instruction is executed at $--IOA (block CB04).

Block CB04, $--IOA: The card read instruction is executed. If the program uses overlap, the instruction is executed in overlap mode. This is done to preserve overlap mode of operation if an overlapped operation is in progress on the other channel. If the instruction were not given in overlap mode, it would not be executed until the I/O operation on the other channel is completed.

Block CB05: A test is made to determine if any I/O channel status indicators are on. If they are all off, control goes to block CB07. If any are on, control passes to the unit record error routine (represented by block CB06).

Block CB06: The reason for entry to the error routine is determined and appropriate action is taken. Operator action may be required. Control returns to block CB07.

Blocks CB07, CB08, and CB09: A test is made to determine if the condition I/O channel status indicator is on. If it is, it indicates an end-of-file condition on the card reader and control is sent to one of three places.

1. EOFADDR specified in the last GET READ, EOFADDR macro used for this file.
2. EOFADDR specified in the DTF for this file if condition 1 does not apply.
3. $--IOA in the file scheduler if neither conditions 1 nor 2 apply. The card read instruction is re-executed (block CB04). In this case, it is NOP'ed and the not ready indicator is turned on. The error routine is entered (block CB06) and a not ready message is typed out indicating that the reader is out of cards. The program enters a wait loop for operator action. If more cards are placed in the reader, the program continues.

Block CB10, $--TRIG: A branch and exit priority alert mode to $ENTRY is executed. There is no significance in the fact that the instruction is a BXPA. It is used as an unconditional branch. I/O operations are started on the channels, if possible, and priority alert mode is entered.

Block CB11: The card count, $--TBC, is incremented by +1.

Block CB12, $--EXIT: Control exits from the file scheduler to the location set by block CB02.

Blocks CB13 and CB14: If the macro is of the format, GET READ TO WORK, the contents of the DTF specified I/O area for this file are moved to the work area specified by the macro. Control returns to the user's program.

UNIT RECORD PUT MACRO AND SCHEDULERS

Block CB21: Before entering the file scheduler on a PUT WORKAREA TO FILE macro, the contents of the macro specified workarea are moved to the DTF-specified I/O area of the printer or punch file. If the macro is of the PUT FILEA TO FILEB format, the current logical record of FILEA is moved to the DTF-specified I/O area of the printer or punch file.

Block CB22, $--FULL: The contents of the B-address register are stored in the I-address of a branch at $--EXIT (block CB29). This initializes the file scheduler's exit.

Block CB23: The record count, $--TBC, is incremented by +1.

Block CB24, $--SFX: A branch and exit priority alert mode to $CS-SFS, the force entry to the channel scheduler, is executed. The channel for which the PUT macro was issued is forced clear of all unchecked I/O operations. This is done to prevent an I/O interlock when the I/O instruction is executed at $--IOA (block CB25).

Block CB25, $--IOA: A punch a card or write a line instruction, depending on the file type, is executed. The data punched or printed is contained in the file's DTF-specified I/O area. If the program uses overlap, the instruction is executed in overlap mode.

Block CB26: A test is made to determine if any I/O channel status indicators are on. If they are all off, control goes to $--TRIG (block CB28). If any indicators are on, control passes to the unit record error routine (represented by block CB27).

Block CB27: The reason for entry to the error routine is determined and appropriate action is taken. Operator action may be required. Control passes to block CB28.

Block CB28, $--TRIG: A branch and exit priority alert mode to $ENTRY is executed. There is no significance in the fact that the instruction is a BXPA. It is used as an unconditional branch. I/O operations are started on the channels if possible and priority alert mode is entered.

Block CB29, $--EXIT: Control exits from the file scheduler to the location set by block CB22.

CLOSE PUNCH SEQUENCE

Block CB31, $--ACT+11: This block is entered from the close sequence for a punch file. The high-order address of the punch file's I/O area is placed in the A-address register by executing a dummy BCE instruction. The contents of the A-address register are stored in the B-address of a move instruction executed in block CB32.

Block CB32: A blank character is moved to the address initialized by block CB31. The B-address of the move just executed is initialized to address the next character of the I/O area by storing the contents of the B-address register, after the move is executed, in the B-address of the move.

Block CB33: A test is made to determine if the address specified by the B-address of the move instruction executed in block CB32 contains group mark/word mark. If it does not, it indicates the end of the I/O area has not been reached and control goes to block CB32 to blank another location. When the end of the area is reached, control passes to block CB34.

Block CB34: A blank card is punched.

Block CB35: A test is made to determine if any I/O channel status indicators are on. If they are all off, control goes to $EXITRU (Chart DF, block 11) to continue the close sequence. If any indicator is on, control passes to the unit record error routine (represented by block CB36).

Block CB36: The reason for entry to the error routine is determined and appropriate action is taken. Operator action may be required. Control then passes to $EXITRU (Chart DF, block 11) to continue the close sequence.

Little Macros

IOCS must provide I/O control other than that afforded by the OPEN, CLOSE, GET, and PUT macros. This additional control is available through the use of the IOCS *little* macros.

These macros enable the programmer to position tape on a unit, to select the pocket into which cards are stacked on card read or punch operations, to position paper on the printer through carriage control, to use the console printer, to read or write records on any tape unit, and to read and write tape labels.

Some restrictions in the use of the *little* macros follow:

The STACK *little* macro may only be used to select cards into specified pockets when the file for which it is issued has a DTF CARDPOC 9 entry.

The RTAPE, WTAPE, and IOBSP *little* macros do not adjust block counts if used for DTF-specified files.

No *little* macros pertaining to tape may be used before the first OPEN macro if there is no DIOCS PRIORITY entry. The reason they may not be used is that the tape error routine is not yet in core storage. The error routine overlays the priority assignment routine during the first OPEN.

A detailed description of how these little macros are executed is presented in the text for Chart CC, which follows.

RTAPE (READ TAPE) AND WTAPE (WRITE TAPE)

Block CC01: If the program uses the overlap special feature, a BXPA to $CS-SFS is executed. The channel for which the macro was issued is cleared of unchecked I/O operations. This block does not exist for the non-overlap situation.

Block CC02: If parameter 4 (fourth operand of the macro) is specified, a word mark is set in the core storage location labeled by parameter 4. This is a switch used by IOCS to indicate to the programmer the availability of the input record (RTAPE) or the output area (WTAPE). The branch instruction at $INTEXT is set to return control to the user's program. Parameter 4 is specified only for overlapped operations.

Block CC03: The format of the I/O instruction is determined by parameters 1, 2, and 3. Parameter 1 specifies the manner (e.g., write tape with word marks) in which the operation is executed, parameter 2 specifies the channel and unit, and parameter 3 specifies the label of the core storage area used. These parameters must be specified for the macro. The I/O instruction is executed. If the operation is read tape and the program does not use overlap, the contents

of the B-address register are stored in an area labeled $ERNOIS at the completion of the read. This initializes the tape error routine's noise length record coding.

Block CC04: If parameter 4 is specified in the macro, it indicates that the operation is to be overlapped and a branch on overlap instruction is executed. If the operation started successfully, control goes to $CS-RET (block CC05). If not, control goes to block CC06 to determine the reason for failure. This block does not exist if parameter 4 is not specified.

Block CC05: The branch at $CS-PR, in the appropriate channel scheduler, is set to go to block CC06. If applicable, channel 2 operation is started and control returns to the user's program in priority alert mode via $INTEXT (set by block CC02). Upon completion of the tape operation, control returns to block CC06 via $CS-PR.

Block CC06: The tape operation is checked by a BEX instruction. The d-modifier of the instruction is 7 if parameter 5 is specified. Otherwise, it is a tape mark. Parameter 5 indicates that the user wishes control to go to his end-of-file/reel routine if the I/O condition channel status indicator is on as a result of the tape operation. Note that wrong length records are not checked. However, IOCS provides the user with information from which he can compute record length if parameter 6 is specified.

Block CC07: The tape error routine corrects the operation if possible and passes control block CC08. Operator action may be required.

Block CC08: If parameter 6 is specified, it indicates the user wishes IOCS to store the contents of the E-(channel 1) or F-(channel 2) address register in an area labeled by parameter 6. An SER or SFR instruction is executed to accomplish the store operation. This block only applies to RTAPE macros.

Block CC09: If parameter 5 is specified, the user wishes control to go to his end-of-file/reel coding on end-of-file or end-of-reel conditions. A BEF instruction, to the user's routine, is executed.

Block CC10: A branch any to itself+1 is executed to satisfy the I/O channel status test requirements. Control goes to one of three places:

1. Parameter 4 is specified	block CC11
2. Not overlap	block CC13
3. Overlap, not parameter 4	block CC14

Block CC11: The word mark in the core location specified by parameter 4 is cleared, indicating to the user the availability of the input record (RTAPE) or output area (WTAPE) for processing.

Block CC12: After channel servicing is completed, control returns to the place of interrupt via $INTEXT. Note that if control dropped through the BOL instruction at block CC04, no interrupt has occurred and $INTEXT is still set to go to the user's program. Control leaves this block in priority alert mode.

Block CC13: If parameter 6 is specified, the contents of $ERNOIS (set by block CC03) are moved to the area labeled by parameter 6. The user may use this for wrong-length-record checking.

Block CC14: I/O operations are started on the channels, if possible, and priority alert mode is restored by the execution of the BEPA instruction at $INTEXT. The branch at $INTEXT, set by block CC02, returns control to the user's program.

CONSL (CONSOLE OPERATION)

Block CC21: If a channel 1 scheduler exists, the channel must be cleared of I/O operations before the console printer is used. A BXPA to $CS1SFS is executed and channel 1 is cleared of all unchecked operations.

Block CC22: The I/O operation specified by parameters 1 and 2 of the macro is executed. Parameter 1 indicates the manner (e.g., read with word marks) in which the console operation is to be performed and parameter 2 is the label of the area in core storage to be used.

Block CC23: A test is made to determine if any I/O channel status indicators are on. If all indicators are off, control goes to block CC25. If any indicators are on, control passes to the console printer error routine (represented by block CC24).

Block CC24: The console printer error routine determines if the busy, data check, I/O condition, or the no-transfer indicators are on. If any are on, the operation must be retried and control returns to block CC22.

Block CC25: If the program uses the overlap feature, control goes to $ENTRY. I/O operations are started on the channels, if possible, and control returns to the user in priority alert mode via the BEPA instruction at $INTEXT. This block is bypassed if the assembly is non-overlap.

OTHER LITTLE MACROS

Other *little* macros are:

RTLBL	(Read Tape Label)
STACK	(Select Stacker and Feed)
IOBSP	(I/O Backspace)
IORWU	(I/O Rewind and Unload)
WTLBL	(Write Tape Label)
SKIP	(Skip Carriage)
IORWD	(I/O Rewind)
IOWTM	(I/O Write Tape Mark)

Block CC31: If the program uses the overlap feature, a BXPA to $CS-SFS is executed to clear the appropriate channel of unchecked I/O operations.

Block CC32: The I/O instruction, specified by the macro parameters, is executed. If the instruction is a

RTLBL in a non-overlap program, the contents of the B-address register are stored in $ERNOIS at the completion of the read operation. This initializes the tape error routine's noise record coding.

Block CC33: A BEX instruction is executed to check the I/O operation. The d-modifier of the instruction is 7 for label operations. It is a group mark for all others. Wrong-length records and end-of-file/reel conditions are not checked on label operations. If none of the indicators tested are on, control goes to block CC35. If any indicator tested is on, control passes to the appropriate error routine (represented by block CC34).

Block CC34: The appropriate error routine (tape or unit record) corrects the operation if possible and passes control to block CC35. Operator action may be required.

Block CC35: This block represents a NOP in the coding of a RTLBL macro. It is used by the tape error routine to determine if the operation which caused entry to it was a label read.

Block CC36: If the macro was a label operation, a branch any to the next sequential instruction is executed to satisfy the I/O channel status test requirements.

Block CC37: If the program uses the overlap feature, control goes to $ENTRY to start channel operations, if possible. Control returns to the user's program in priority alert mode via a BEPA at $INTEXT.

IOSYS FORCE (CLEAR CHANNELS)

Block CC41: A BXPA to $CS1SFS is executed if the operand is FORCE, or FORCE with 1 specified as the second or third parameter. Channel 1 is cleared of unchecked I/O operations. This block does not exist if there is no channel-1 scheduler.

Block CC42: A BXPA to $CS2SFS is executed if the operand is FORCE, or FORCE with 2 specified as the second or third parameter. Channel 2 is cleared of unchecked I/O operations. This block does not exist if there is no channel-2 scheduler. Control returns to the user's program.

IOSYS RESUME (START CHANNELS)

Block CC51: If the program uses the overlap feature, a BXPA to $ENTRY is executed. Channel operations are started, if possible. Control returns to the user's program in priority alert mode via the BEPA instruction at $INTEXT.

PSTAC (SELECT PUNCH STACKER)

Block CC61: The I/O instruction in the file scheduler specified by parameter 2 is set to stack cards in the pocket specified by parameter 1. This is accomplished by an instruction which moves parameter 1 to $--IOA+3, where -- is parameter 2.

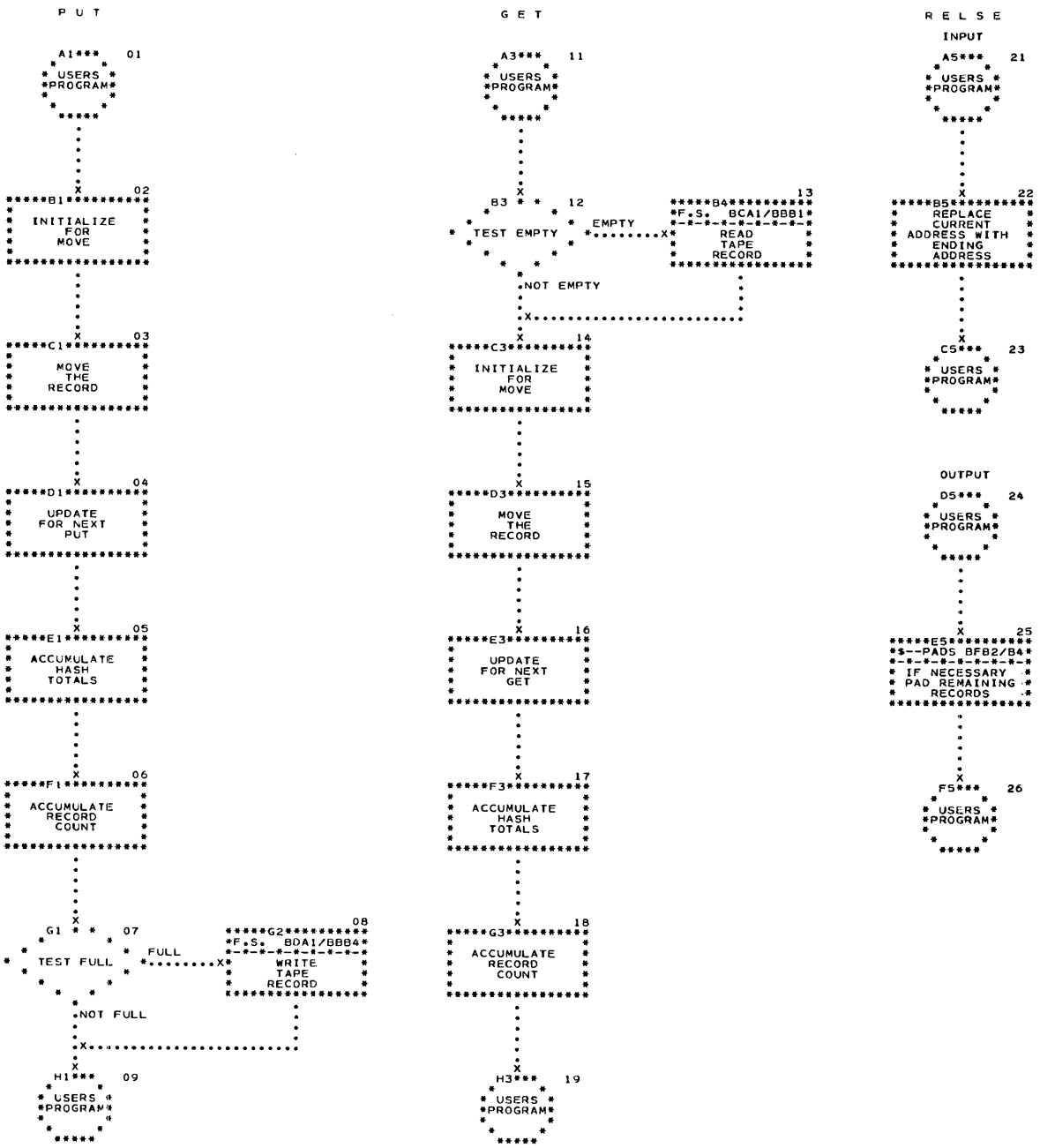


Chart CA. PUT, GET, and RELSE Macros

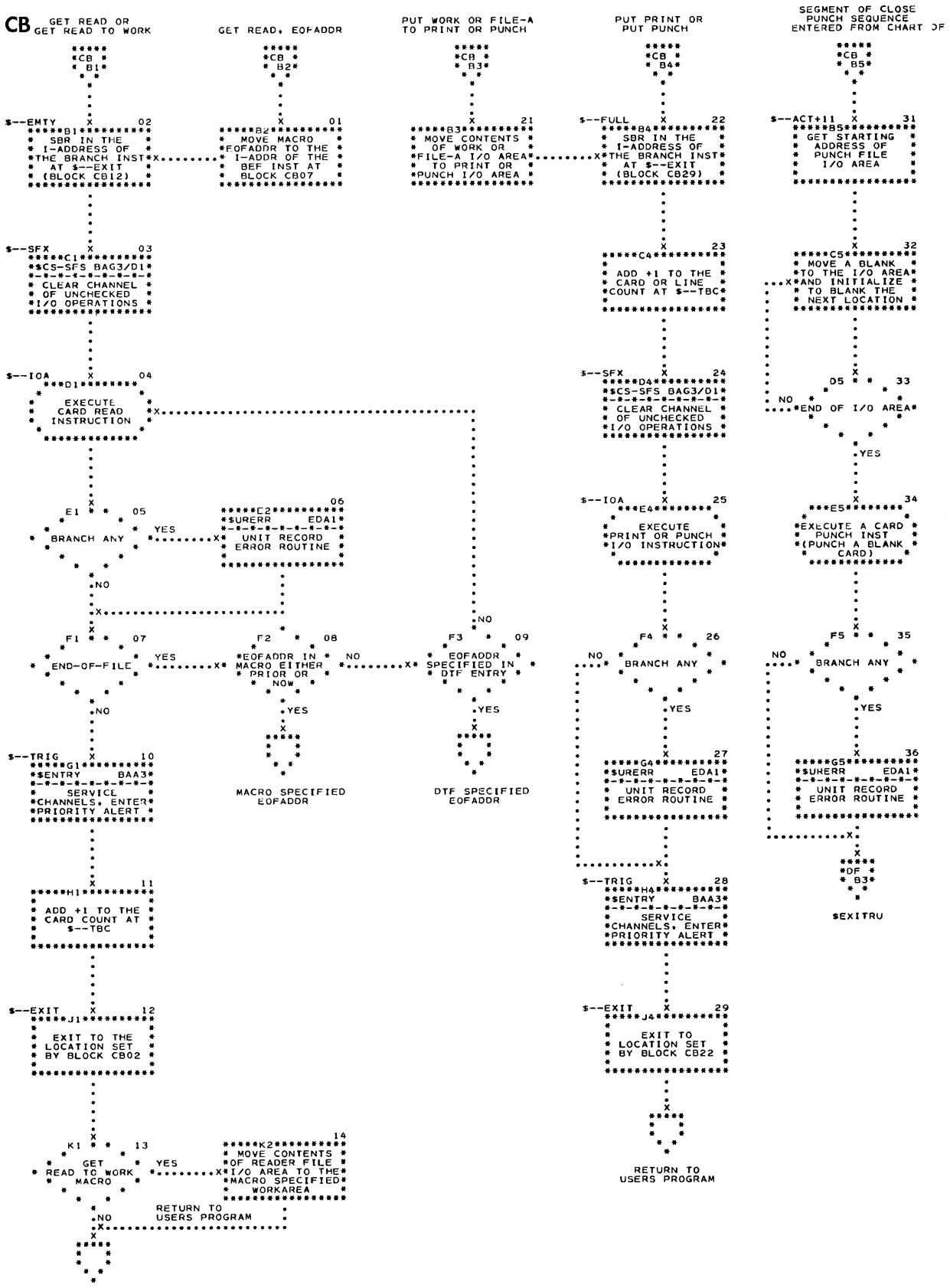


Chart CB. UR GET/PUT Macros and Schedulers

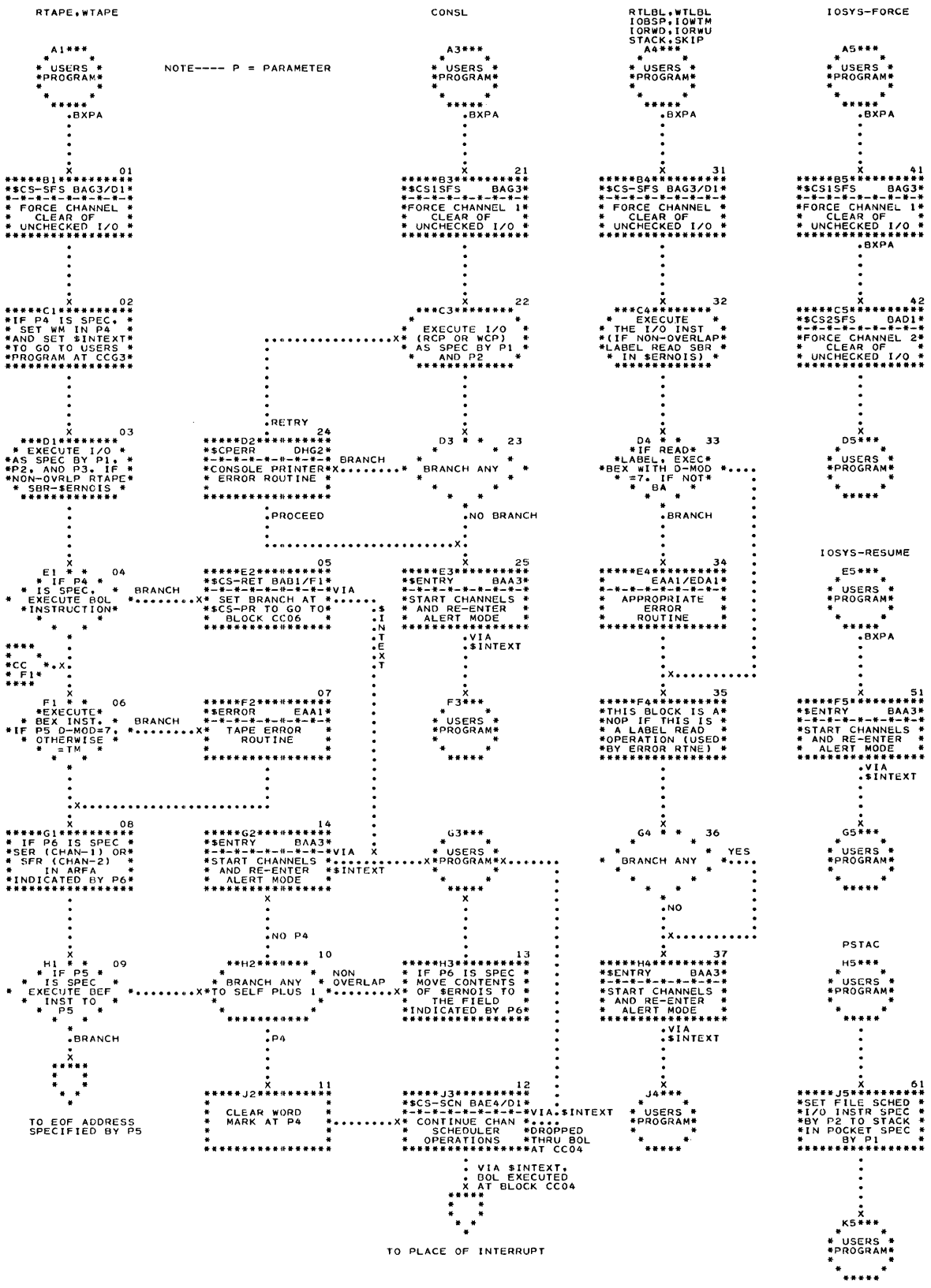
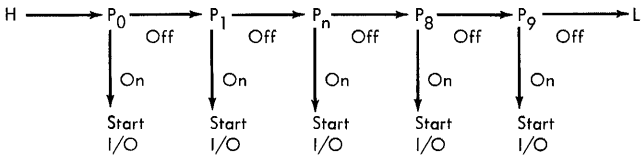


Chart CC. Little Macros

Open, Close, and End-of-Reel Procedures

Priority Assignment Routine

File priority allows the user to determine the order in which non-forcing I/O requests for two-area tape files on a channel are serviced. Such requests are *interrogated* in the pending switch network. This network is an aggregate of the linkages of the respective file scheduler pending switches in their relative priority order (high to low), and the linkage from the channel scheduler to the highest-priority pending switch, and the exit linkage from the lowest-priority pending switch (Figure 2).



H is the instruction labeled $CS-S3 in the associated channel scheduler.
P_n is the instruction representing a pending switch of priority n.
L is $CS2ENT for channel 1, $INTEXT for channel 2.

Figure 2. Pending Switch Network

A non-forcing I/O request for a file is *recorded* during scheduling operations by the file scheduler when an area is empty (input) or full (output); however, only the request of highest priority on the channel is *honored* at a time through the channel scheduler entry to the pending network.

PRIORITY ASSIGNMENT—DEFINITION AND USAGE

Assignment of file priorities is the process of ordering the pending switch network to the priority order desired. The 1410 IOCS allows the user to set the priority order at source or object time according to the DIOCS PRIORITY option chosen:

DIOCS PRIORITY *Option*	*Results*
NONOVERLAY	A priority assignment routine is generated and remains in storage at all times. Priorities are assigned during OPEN and re-OPEN operations for all two-area files named in the OPEN macro and for those not named but active.
ASSEMBLE	A priority assignment routine is not generated. The assignment of file priority is made at compilation time; the priority order is determined by the sequence of source DTF cards. The priority order may not be modified at object time.
Omission of DIOCS PRIORITY	A shorter priority assignment routine is generated, which is overlayed immediately following its execution. File priorities are assigned only during the first OPEN operation and may not be modified later. A two-area file opened subsequently is treated as a one-area file.

PRIORITY ASSIGNMENT—METHODOLOGY

The priority assignment routine builds a table of pending switch addresses, each address being a DCW defining the low-order location of a pending switch instruction. A two-character DCW, representing the priority and channel-overlap identification for the file, precedes

	Pending Network	Portion of Table		Explanation
As Compiled	H → L		bb	
			bbbbb	
			bb	Chan 2 dummy arg
			DCW $CS2S3+5	Chan 2 sched entry to pend net
			DC @b	Table arg for H
			DCW $CS1S3+5	H (low-order)
		$PARG	DCW bb	Search arg position
Step A	H → L P_7 → L		bb	
			bbbbb	
			bb	
			DCW $CS2S3+5	Function from look-up (SBR X15)
			DC @b	
			DCW $CS1S3+5	
		$PARG	DCW @7	Search arg for P_7
Step B			bb	
			DCW $CS2S3+5	Table moved 7 positions down
			DC @b	
			DCW $CS1S3+5	H re-inserted from saved loc
			DC @7	Table arg for P_7
			DCW xxxxx	P_7 (low-order)
		$PARG	DCW @7	Search arg for P_7
Step C	H → P_7 →		L	

Figure 3. Table of Pending Switch Addresses

each address. Each pair in the table act as function and table argument for table look-up operations; the starting location in the table is low priority for channel 1.

The assigning of file priorities is done on a file-by-file basis and the assignment of each file is complete:

1. The file is *inserted* into its relative channel-priority order in the table.
2. The file is *inserted* into its relative priority order in the pending switch network for the channel.

This methodology is illustrated in Figure 3 for the case of the first two-area file at the time of the first OPEN operation. PRIORITY overlay and a file priority of 7 on channel 1 are assumed.

Description of Priority Assignment

Block DA01, $PAHSK: (Overlay) Because the linkage table routine causes control for all files named in the OPEN macro instruction to go to the priority assignment routine (due to the 9 in the all-inclusive sequence at $POX in the linkage table described in Chart DF, block 08), a file type test is made to determine if the file is two-area tape. If it is not a two-area tape file, control branches to $PAEXIT, block DA25. For two-area tape files, control passes to block DA02.

(Non-overlay) Because the linkage table routine directs only two-area tape files to go to the priority assignment routine, block DA01 does not exist.

Block DA02, $PAHSK: The file is two-area tape. The operand of the $PENSWE indexed label entry ($--WTG) is moved to the B-address of the $OUT instruction, block DA16, and a +44 is added to it. The resulting address, $--WTG+44, points to the low-order location of the file's pending switch instruction.

Block DA03: The priority code (indexed label, $DTFACT-1) and the channel identification (indexed label, $DTFACT+4) for the file are moved to $PARG. $PARG is a two-position DCW used as search argument for the look-up operation in the table of pending switch addresses.

1. Non-overlay: Control then proceeds to the block sequence DA04 through DA12 for processing in case the file is being re-opened.
2. Overlay: Control proceeds directly to block DA13; blocks DA04 through DA12 do not exist. File priorities are sorted only during the first OPEN operation, since the priority assignment routine is overlaid immediately following its execution for all the files named in the first OPEN.

PROCEDURE IN CASE OF RE-OPEN (NON-OVERLAY ONLY)

Block DA04: A SBR into index register 15 is executed to point to the first pending switch address in the table (first table function).

Blocks DA05, $COMP, DA06, and DA07: A loop is performed to determine if the file's pending switch address (contained in the B-address of $OUT) is in the table. A test for blank table argument (0-6+X15) at block DA05 precedes the accessing of the next function (SBR into X15) at block DA06 and the comparison of the previous function (7+X15) to the file's pending switch address performed at block DA07. A blank table argument means the file is not represented in the table and control branches to $END, block DA13.

Blocks DA08 through DA12: The pending switch is eliminated from the file scheduler pending network on the channel and from the table. (It will later be reinserted according to its new priority.)

At blocks DA08 and DA09, the contents of the function which represent the next-higher priority pending address (0+X15) and the contents of the previous function which represent the file's pending address (7+X15) are moved to the A- and B-addresses, respectively, of $LINK. At $LINK, block DA10, the file's pending instruction is moved to overlay the higher-priority pending instruction, thus eliminating the file from the network.

At blocks DA11 and DA12, a loop is performed, successively overlaying each table function (starting with file's 7+X15) with the next-higher priority function (0+X15). After each move, the next function is accessed (SAR into X15) and the table argument (6+X15) is tested for non-blank. When a blank table argument is reached, control proceeds to $END at block DA13.

COMMON PROCEDURE TO INSERT FILE INTO PENDING NETWORK AND TABLE

Blocks DA13, $END, and DA14: A look-up low or equal with $PARG as search argument is performed in the table of pending switch addresses. The look-up is followed by an SBR into X15 instruction. Note that the function addressed by index register 15 has a *2nd-higher* than file priority on the respective channel.

Block DA15: The A-address of $OUT is initialized with the next-higher priority pending switch address (7+X15).

Block DA16, $OUT: The file is partially inserted into the pending network by overlaying the branch address of the pending switch to be inserted with the address of the higher-priority pending switch. (this corresponds to Figure 3, step A.)

Blocks DA17 through DA20: The file is now *inserted* into the table in its relative channel-priority position. (This corresponds to Figure 3, step B.)

At block DA17, a MRCWG moves the entire table down 7 positions, starting at lower storage, up to the next-higher priority pending address (where a high-order group mark has been placed). At blocks DA18 through DA20, chained MLC commands overlay the next-higher

priority pending address (7+X15) by the file's pending address (from the B-address of $OUT), move $PARG to the *vacated* table argument (2+X15), and overlay the *vacated* 2nd-higher priority address (0+X15) by the 1st-higher priority pending address (from the A-address of $OUT).

Blocks DA21, DA22, and DA23: The insertion of the file into the pending network, started at block DA16, is now completed by overlaying the branch address of the 1st-higher priority pending instruction to point to the new file's pending switch. (This corresponds to Figure 3, step C.)

At blocks DA21 and DA22, the B-address of $IN, block DA23, is initialized with the 1st-higher priority pending address (0+X15) and +6 is subtracted from the file's pending address at the B-address of $OUT (to adjust the address to high-order).

At $IN, 5 chained MLCS commands move the adjusted B-address of $OUT to the branch address of the 1st higher priority pending switch instruction.

Block DA24: If the non-overlay option is in effect, blocks DA25 through DA28 do not exist. After the file reference address for the file is restored to index register 15, control branches to $ENTA (block DB01) to start open procedures for the file.

Blocks DA25 and DA26: If the overlay option is in effect, block DA24 does not exist. After +6 is added to the file counter, $COUNT, a test is made here to determine whether the next check character in the calling sequence work area (at $DTFBX+1) is the terminal check character, J. If not, there is another file to process and control branches to $EXIT, block DF03.

Blocks DA27 and DA28: All files have been handled by the priority assignment routine. The contents of $COUNT (which contains the count for all files handled, (i.e., 6 times n, where n is the number of files) are subtracted from the A-address of $EXIT (the file-accessing instruction, block DF03) so that $EXIT is reinitialized to point to the beginning of the macro calling sequence. The following specific tape sequence is then moved in over the all-inclusive sequence at $POX in the routine linkage table. The all-inclusive sequence is described at block DF08.

Tape Sequence

```
        DCW     $TPCLOS
        DC      @ J@
        DC      $ENTA
$POX    DC      2
```

Control now branches unconditionally to the load program at location 00281 to bring in the remaining IOCS routines to overlay the priority assignment routine. The load execute is to $EXIT, block DF03 in the linkage table routine, where the files are now processed according to file type.

Description of Open Procedures

Blocks DB01, $ENTA, and DB02: The base channel and unit identification for the file (indexed labels, $AD-6 and $AD-8, respectively) are moved to $ARG. A look-up (LLE) is then executed in the table of file reference addresses with $ARG as the search argument. The format of the table is shown in Figure 5.

Block DB03: The contents of index register 15, the file reference address for the file, are moved to the table function.

Block DB04: The various internal count fields (e.g., record count, if specified) are blanked out.

Block DB05: The address of the file scheduler initialization coding block for the file, (indexed label, $DTF1), is moved to $FINIT, block DB09. For applications not specifying CHANNEL CHANGE, blocks DB06 through DB08 do not exist and control passes to block DB09. Blocks DB06 through DB08 are executed for CHANNEL CHANGE applications.

Block DB06: Two dummy clear word mark instructions are executed to initialize the A- and B-address registers. The A-address register is set to point to $CCTAB+27 in the channel change table. The B-address register is set to point to indexed label $AB-7. The contents of the A-register are placed in index register 15.

Block DB07: A branch if bit equal instruction (with an A d-modifier) is used to test $AD-7 for a channel 2 file indicator. (An @ indicates channel 1, an * channel 2.) If the file is a channel 2 file (the B bit in the A character matched the B bit in the *), control branches directly to the common exit, block DB09. For a channel 1 file, control passes to block DB08.

Block DB08: It is a channel 1 operation. A dummy clear word mark instruction is executed to initialize the A-address register to point to $CCTAB+1 in the channel change table. The contents of the A-register are placed in index register 15.

Block DB09, $FINIT: At $FINIT, control exits to the file scheduler initialization coding block as set by block DB05. Block DB10 represents this initialization block.

Block DB10: The file scheduler is initialized according to the file characteristics (e.g., file type, number of areas, and WLR procedure). A table of initialization function performed versus the file characteristics is presented in Chart BE. Control passes to $ENTAB, block DB11.

Block DB11, $ENTAB: $ENTAB is the place of return after file scheduler initialization or after a standard header label error when a retry is desired. Index register 15 is reinitialized with the file reference address from $DTFBX in the calling sequence work area. When DIOCS CHANNEL CHANGE is in effect, this block does not

exist and $ENTAB becomes the label of the next block (DB12).

Blocks DB12 and DB13: The rewind indicator for the file (indexed label, $DTFL5) is tested. If no rewind is specified (code=0), control branches to block DB14. Otherwise, control passes to the utility subroutine at $RWDRU to rewind the tape. Control returns from the subroutine to block DB14.

Block DB14: The label type indicator (indexed label, $DTFLB) is tested. If it is a standard label file (code =0), control branches to $ENTC on Chart DC, block 01. The return after standard header-label procedures is to $ENTD, block DB16, for input, or $ENTF, block DB27, for output.

Block DB15: The file type code (indexed label, $DTFL1) is tested. A 1 indicates an input file; control passes to $ENTD, block DB16. A 0 indicates an output file; control passes to $ENTF, block DB27.

Block DB16: It is an input file (code=1). The user's Exit 7 address (indexed label, $E7) is moved to the exit routine (at $SWBXA) in case the file uses Exit 7.

Block DB17: The Exit 7 indicator for the file (indexed label, $D7) is tested. If Exit 7 is not used, control branches to block DB21.

Block DB18: The exit routine at $SWBX executes an SBR which sets the return address, block DB21, into the re-entry routine (at $STLEXT). Index register 15 is restored for the user, and control branches (as set by block DB16) to the Exit 7 address.

Block DB19: Exit 7 can be used for checking of input header labels in lieu of, or in addition to, standard label processing. To process an additional label, the user must give a RTLBL macro instruction. For non-standard labels, the user must give a RTLBL macro instruction for each label before processing.

Block DB20: The re-entry routine is executed at $REENT to save index register 15 for the user, to restore it for IOCS (from $DTFBX), and to branch to block DB21 (set by block DB18).

Blocks DB21 and DB22: An indicator (indexed label, $DTFL4) is tested to determine if the file has a tape mark after the header. If so (code=1), the subroutine at $READRU, block DB22, is executed to bypass the tape mark. If there is no tape mark, (code=0), control passes to block DB23.

Blocks DB23, $ENTI, and DB24: $ENTI is a common gathering point for an input or output file in OPEN, FEORL, and end-of-reel operations. If DIOCS CHECKPOINT is in effect, the $CHKPT subroutine, block DB24, is executed to take a checkpoint on the specified checkpoint tape.

Blocks DB25, $ENTJ, and DB26: $ENTJ is a common coding block executed for an input or output file in OPEN, CLOSE, FEORL, and end-of-reel operations. The file type indicator (indexed label, $DTFACT) is tested for number of areas. If it is two-area (code=2), the pending switch for the file is set off at block DB26. If the file is not a two-area file, control passes to $EXITRU, on chart DF at block 11. $EXITRU is the termination routine which tests if there is another file to process.

Block DB27, $ENTF: It is an output file (code=0). The user's Exit 5 address (indexed label, $E5) is moved to the exit routine (at $SWBXA) in case the file uses Exit 5.

Block DB28: The Exit 5 indicator for the file (indexed label, $D5) is tested. If Exit 5 is not used, control branches to block DB32.

Block DB29: The exit routine at $SWBX executes an SBR which sets the return address, block DB32, into the re-entry routine at $STLEXT. Index register 15 is restored for the user, and control branches (as set by block DB27) to the Exit 5 address.

Block DB30: Exit 5 can be used for the writing of output header label(s) in lieu of, or in addition to, standard label processing. For standard labels, the user can build the label in the IOCS label area and give a WTLBL macro instruction, to write his additional label. For non-standard labels, the user must give a RTLBL macro instruction if he wishes to check the output label and, after checking, execute an IORWD macro to rewind the tape before building his label.

Block DB31: The re-entry routine at $REENT saves index register 15 for the user, restores it for IOCS (from $DTFBX), and branches to block DB32 (set by block DB29).

Blocks DB32 and DB33: An indicator (indexed label, $DTFL4) is tested to determine if a tape mark is to be written following the header label(s). If so (code = 1), the utility subroutine at $WTMRU, block DB33, is executed to do so. Otherwise, or after the tape mark is written, control branches to $ENTI, block DB23.

Description of Standard Header Label Procedures

Block DC01, $ENTC: The file type indicator (indexed label, $DTFL1) is tested. If it is an output file (code = 0), control proceeds to block DC26. If it is an input file (code = 1), control goes to block DC02.

Block DC02, $ENTCR: The header label is read into the IOCS label area. For an output file, this block is entered only when the test at block DC27 indicates the label is to be checked (DTF CHECKLABEL IDENT OR ALL).

Block DC03: The label check indicator for the file (indexed label, $DTFL3) is tested. If no check is to be made (code = 1), control leaves label procedures to continue input processing on Chart DB at block 16. For an output file, control cannot exit (see block DC27).

Block DC04: The label identifier is compared to 1HDRb. Unless they match, the label is non-standard or

there is no label. A branch unequal is made to $NHL, block DC23, to process the error.

Block DC05: The file type indicator for the file (indexed label, $DTFL1) is tested. If it is output (code = 0), control branches to $OPHD, block DC14.

Block DC06: It is an input file. The header label fields (included by the DIOCS and specified by the DTF for the file) are compared to their file reference counterparts. Assuming DIOCS LABELDEF CHECK:

1. If the user specified IDENT in the DTF CHECKLABEL card, only the file name is checked.
2. If the user omits the DTF card, SERIALNUM, the file serial is not checked.

Blocks DC07 and DC08: The fields compared do not match. An identifying message, 30122bFILbxxxn, where xxxn represents the base tape identification, is set up in the console message area, and the $NOTE subroutine (represented by block DC08) is executed to type the message.

Blocks DC09 and DC10: The label fields in positions 11-35 of the IOCS label area (see Figure 4) are set up in the console message area, and the $NOTE subroutine (represented by block DC10) is executed to type the message.

Block DC11: The internal fields corresponding to the fields in the message typed at block DC10 are set up in the console message area.

Block DC12, $NIH: The $HALT subroutine is executed to type either the message set up by block DC11 or by block DC23, and to enter a waiting loop for operator reply.

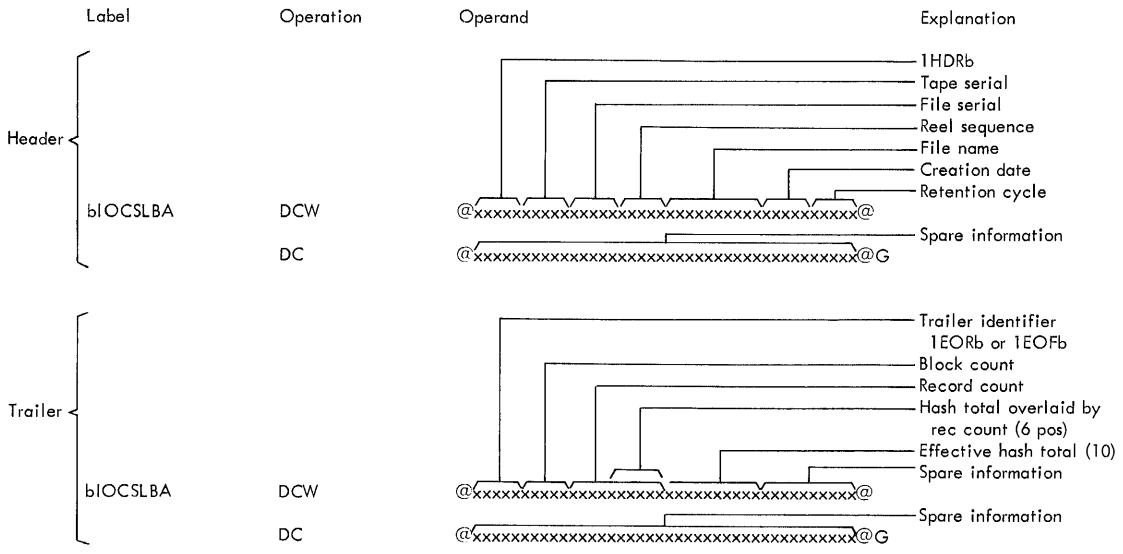
Block DC13: The first character in $REPLY, the input area for the operator's reply, is tested. If it is an A (indicating the ACCEPT option was chosen), the input header error is ignored and control branches to $ENTD, block DB16, to continue input processing. Otherwise, the operator has replaced the incorrect input tape reel with the proper one and control branches to $ENTAB, block DB11, to prepare to process the new label.

Block DC14, $OPHD: It is an output file. The retention cycle is added to the creation date in the IOCS label area.

Blocks DC15 and DC16: If the sum exceeds 366, 635 is added to the sum in order to obtain the actual retention date.

Block DC17: If today's date (in core locations 00115-00119) is beyond the retention date, the tape can be used and control branches to $OPHDAB, block DC28, to assemble the output header.

Block DC18, $DATER, and Block DC19: Today's date lies within the retention date. An identifying message, 40131bDATbxxxnbyyddd, where xxxn is the base tape identification and yyddd is the computed retention date, is prepared in the console message area.



● Figure 4. Header and Trailer Formats in Storage

Block DC20, $NOH: The $HALT subroutine is executed to type either the message set up by blocks DC18 and DC19 or by block DC25, and to enter a waiting loop for operator reply.

Blocks DC21 and DC22: The first character in $REPLY is tested.

1. If an A (indicating the operator chose the ACCEPT option), the output header error is ignored and control branches to $OPHDAB, block DC28, to assemble the new output header.
2. If numerical, $REPLY contains the correct day's date inserted by the operator. The correct date is moved into core locations 00115-00119 and control branches to $ENTAB, block DB11, to prepare to reprocess the output header label.
3. If neither, the operator has replaced the incorrect output tape reel with the proper one and control branches to $ENTAB, block DB11, to prepare to reprocess the new label.

Block DC23, $NHL: It is not a standard header label. An identifying message, 30133bNIHbxxxn, where xxxn is the base tape identification, is prepared in the console message area.

Block DC24: The file type indicator (indexed label, $DTFL1) is tested. If it is an input file (code=1), control branches to $NIH, block DC12.

Block DC25: It is an output file (code=0). The first part of the message is altered to so indicate. The message is now, 40130bNOHbxxxn. Control passes to block DC20 to type the message.

Block DC26: The test at block DC01 indicates it is an output file. The Exit 3 indicator (indexed label, $D3) is tested. If this file uses exit 3 (code=1), control goes to block DC41; IOCS checking of the header label is bypassed.

Block DC27: The label check indicator for the file (indexed label, $DTFL3) is tested. If the output label is to be checked (code=0), control goes to $ENTCR, block DC02. Otherwise (code=1), control proceeds to block DC28.

ASSEMBLE STANDARD OUTPUT HEADER LABEL

Block DC28, $OPHDAB: $OPHDAB is reached directly from block DC27, if no output label checking is to be done for the file, or via connector A4 (from block DC17 or DC21), after the output label has been checked. The indicator (indexed label, $D9) is tested to see if the file serial is to be made equal to the tape serial of the first reel of the file. If not (code=0), control branches to block DC31.

Block DC29: The check character in the calling sequence work area ($DTFBX-5) is tested for an * to distinguish between an OPEN and end-of-reel operation. If it is end-of-reel (*), control branches to block DC31.

Block DC30: It is the first reel of the file (OPEN operation). The tape serial in the IOCS label area is moved into the file serial field for the file (indexed label, $IFS).

Block DC31: Today's date in core locations 00115-00119 is moved to the creation date field for the file (indexed label, $ICD).

Block DC32: The last 40 positions of the IOCS label area are blanked out.

Block DC33: After the various header fields for the file are chain-moved into the IOCS label area with word marks (see Figure 4, positions 11-40), the identifier 1HDRb is moved to positions 1-5.

Block DC34: The user's Exit 4 address (indexed label, $E4) is moved to the exit routine (at $SWBXA), in case the file uses Exit 4.

Block DC35: The Exit 4 indicator (indexed label, $D4) is tested. If this file does not use Exit 4 (code= 0), control branches to block DC39.

Block DC36: The exit routine at $SWBX executes an SBR which sets the return address, block DC39, into the re-entry routine (at $STLEXT). Index register 15 is restored for the user, and control branches (as set by block DC34) to the Exit 4 address.

Block DC37: Exit 4 can be used for altering the first 40 header positions in the label area and/or entering additional information into the last 40 positions.

Block DC38: The re-entry routine (at $REENT) saves index register 15 for the user, resets it for IOCS (from $DTFBX), and branches to block DC39 (as set by block DC36).

Block DC39, $ENTE: The utility subroutine is executed at $RWDRU to rewind the output tape.

Block DC40: The subroutine at $WRITRU is executed to write the assembled header label on the tape. Control then exits to continue processing the output file at $ENTF, block DB27.

Block DC41: The user's Exit 3 address (indexed label, $E3) is moved to the exit routine (at $SWBXA).

Block DC42: The exit routine at $SWBX executes an SBR which sets the return address, block DC39, into the re-entry routine (at $STLEXT). Index register 15 is restored for the user, and control branches (as set by block DC41) to the Exit 3 address.

Block DC43: Exit 3 can be used for assembling a label in the IOCS label area. This label is in lieu of IOCS label-building (blocks DC28 through DC33). If there is a label on the tape and the user wishes to check it, he must issue a RTLBL before he can check it and assemble the new label.

Block DC44: The re-entry routine (at $REENT) is executed to save index register 15 for the user, reset it for IOCS (from $DTFBX) and branch to block DC39 (as set by block DC42).

Description of End-of-Reel Procedures

Block DD01, $EORU: End-of-reel processing shares many of the routines and coding blocks used for OPEN and CLOSE macro processing. These macros furnish the information needed for processing by means of the calling sequence. An end-of-reel operation is processed as a one-file calling sequence operation. The reader should refer to the description of the linkage routine at blocks DF01 through DF14. An SBR into the A-address of $EXIT, block DF03, furnishes the termination routine (which will be entered on completion of the end-of-reel operation) with the address of the necessary termination character (see block DF11). This address, which is the file reference address, also serves to furnish the termination routine with the linkage pivot back to the file scheduler (see blocks DF13 and DF14). An SBR into $DTFBX is executed to set the calling sequence work area with the file reference address, with which index register 15 will be initialized (see block DD02). An * is moved into the calling sequence work area at $DTFBX-5 to serve as the defining check character for an end-of-reel operation.

Block DD02: The clear channels routine is executed. Both the channels are cleared of unchecked I/O operations, and IOCS is set to ignore start channel operations. After the contents of index register 15 are saved for the user, it is set with the file reference address (from $DTFBX).

Block DD03: The file type indicator (indexed label, $DTFL1), is tested to distinguish between an input and an output operation. If the latter (code = 0), control branches to $OPEOR, block DD25.

INPUT TRAILER PROCEDURES

Block DD04: The A-address of $EXIT, which was set at block DD01 to point to the file reference address, is incremented by 7 to serve as the input file's termination character address and linkage pivot return to the file scheduler.

Block DD05: The label code for the file (indexed label, $DTFLB) is tested. For standard labels, control goes to block DD06; for non-standard, it goes to block DD23.

Block DD06, $IPEOR: It is a standard label file (code = 0). The subroutine, $READRU, is executed to read the trailer label into the IOCS label area.

Block DD07: An identifying message, 10134bTIEbxxxn, where xxxn is the base tape identification, is prepared in the console message area, in case of a trailer label count discrepancy.

Block DD08: Word marks are set into the IOCS label area according to the length of the fields to be compared.

Blocks DD09 and DD10: The count fields (if any) specified by DIOCS COUNTS (HASH and/or RECORD) and the block count are, in turn, compared to the corresponding trailer label fields.

Blocks DD11: The $NOTE subroutine is executed to type the identifying message set up by block DD07.

Blocks DD12 and *DD13:* The counts in the IOCS label area are moved into the console message area, and the $NOTE subroutine is executed to type them.

Blocks DD14 and *DD15:* The internal counts for the file are moved into the console message area with the left-most six positions of the hash total field, if specified, truncated (see Figure 4). The $NOTE subroutine is executed to type the message.

Block DD16, $BCRT: The user's Exit 6 address (indexed label, $E6) is moved to the exit routine (at $SWBXA), in case the file uses Exit 6.

Block DD17: The Exit 6 check indicator (indexed label, $D6) is tested. If this file does not use Exit 6 (code = 0), control branches to block DD21.

Block DD18: The exit routine at $SWBX executes an SBR which sets the return address, block DD21, into the re-entry routine (at $STLEXT). Index register 15 is restored for the user, and control branches (as set by block DD16) to the Exit 6 address.

Block DD19: Exit 6 can be used for various purposes depending on the characteristics of the input file. For a standard label file, it can be used to read and process additional trailer labels. For a non-standard label file, it can be used to read and process the trailer label(s). In the latter case, the user must test for end of file. If it is, the user can get to his end-of-file address by moving an F to $LBA+3 in the IOCS label area to overlay the R (see block DD24). Note that if the user branches directly to his end-of-file address, start channel operations will be ignored until another end-of-reel or macro has been processed; also, there is a possibility that the file may start an I/O operation.

Block DD20: The re-entry routine (at $REENT) saves index register 15 for the user, resets it for IOCS (from $DTFBX), and branches to block DD21 (as set by block DD18).

Block DD21: The fourth position in the label identifier field in the IOCS label area ($LBA+3) is compared to an R. If it is an R, a branch-equal is made to $ENTH, block DE28, to continue input end-of-reel processing.

Block DD22, $ENTG: This block is reached either from block DD21 when it is end of file (F) or from block DD23 for a non-standard label file that does not use Exit 6. The user's end-of-file address (indexed label, $DTFA) is moved to the A-address of $EXIT, to substitute this address as the terminal address for use of the termination routine. Control then goes to the termination routine via $ENTJ. Note that since the linkage to the file

scheduler at $EXIT is destroyed, the user cannot return to IOCS. The user must use Exit 6 when processing a multi-reel, non-standard file.

Blocks DD23 and DD24: It is a non-standard label file (see block DD05). The Exit 6 indicator for the file (indexed label, $D6) is tested. If Exit 6 is not used, the file is treated as if it is at end of file and control branches to $ENTG, block DD22. Otherwise, the IOCS label area is made to appear as end of reel by placing 1EORb into its identifier field for the proper operation of blocks DD16 through DD21.

Block DD25, $OPEOR: It is an output end of reel. The Exit 8 check code for the file (indexed label, $D8) is tested. If Exit 8 is not used, control branches to $OPEORA, block DE05.

Block DD26: The operand of the SAR instruction at block DD28 is set with the contents of $TRIGEN (indexed label-file reference address + 18) in case a two-area file is being used.

Block DD27: The file type code (indexed label, $DTFACT) is tested for number of areas. Control passes to block DD30 for a one-area file.

Block DD28: It is a two-area file. The contents of the A-register, which is the I-address of the instruction at block DD27, are stored into the branch address of $--TRIG (see block DD26). Control branches to $--WTG in the file scheduler (represented by block DD29).

Block DD29: At $--WTG, the pending switch for the file is tested. If it is on, the last block is written on tape by a force operation. If it is off, or after the force, control returns to block DD30 from $--TRIG (as set by block DD28).

Block DD30: The user's Exit 8 address (indexed label, $E8) is moved to the exit routine (at $SWBXA).

Block DD31: The exit routine at $SWBX executes an SBR which sets the return address $OPEORA, block DE05, into the re-entry routine (at $STLEXT). Index register 15 is restored for the user, and control branches (as set by block DD30) to the Exit 8 address.

Block DD32: Exit 8 requires special programming to tailor its use for normal user requirements. It is recommended that the user make use of Exits 1 and 2 in lieu of Exit 8 for these purposes.

Block DD33: The re-entry routine (at $REENT) saves index register 15 for the user, resets it for IOCS (from $DTFBX), and branches to $OPEORA, block DE05 (as set by block DD31).

Description of Close Procedures

Block DE01, $TPCLOS: The contents of $DTFA (indexed label) are moved to the I-address of the instruction at block DE02 to handle the case of a blocked output file.

Block DE02: The file type code (indexed label, $DTFL1) is tested. If it is an output file (code = 0), control branches to the address as set by block DE01. This address is to $--PADS in the file scheduler (represented by block DE03) for a blocked file, or to $CLSABA, block DE04 (since padding does not apply) for an unblocked file. If it is an input file (code = 1), control goes to block DE04.

Block DE03: The block is padded and the file is put on pending, if necessary.

Block DE04, $CLSABA: The check character in the calling sequence work area ($DTFBX−5) is checked for an * to distinguish between a FEORL and a CLOSE operation. If the character is an *, the operation is FEORL and control passes to block DE05; otherwise, control goes to block DE06.

Block DE05, $OPEORA: It is an FEORL operation, or an output end-of-reel operation (from block DD25). The label identifier 1EORb is moved to the IOCS label area identifier field. Control passes to block DE08.

Blocks DE06 and DE07: It is a CLOSE operation. The label identifier 1EOFb is moved to the IOCS label area identifier field. The reel sequence field for the file (indexed label, $HRS−1) is zeroed.

Block DE08, $CLSA: The file type code (indexed label, $DTFL1) is tested to distinguish between an input and an output operation. If it is an input file (code = 1), control goes to block DE28.

OUTPUT TRAILER PROCEDURES

Block DE09: The operand of the SAR instruction at $PRIME3, block DE11, is set with the contents of $TRIGEN (indexed label-file reference address +18) in case it is a two-area file.

Block DE10: The file type code (indexed label, $DTFACT) is tested for number of areas. For a one-area file, control passes to $PRIMER, block DE13; for a two-area file, to $PRIME3, block DE11.

Block DE11, $PRIME3: It is a two-area file. The contents of the A-register, the address $PRIMER (from the I-address of the instruction at block DE10), are stored into the branch address of $--TRIG (see block DE09). Control branches to $--WTG in the file scheduler (represented by block DE12).

Block DE12: At $--WTG the pending switch for the file is tested. If it is on, the last block is written on tape by a force operation. If it is off, or after the force, control returns to $PRIMER, block DE13, from $--TRIG (as set by block DE11).

Block DE13, $PRIMER: A tape mark is written on the output tape by the utility subroutine at $WTMRU.

Block DE14: The label indicator for the file (indexed label, $DTFLB) is tested. If it is a standard label file (code = 0), control proceeds to the block sequence

beginning at $CLSB, block DE15. Otherwise (code = 1), control goes to $SUSXA, block DE22.

Block DE15, $CLSB: After the last 50 positions of the IOCS label area are blanked, the various internal counts are moved into the first 30 positions as shown in Figure 4. The trailer identifier was moved at block DE05 or DE06.

Block DE16: The user's Exit 1 address (indexed label, $E1) is moved to the exit routine (at $SWBXA), in case this file uses Exit 1.

Block DE17: The Exit 1 check indicator is tested. If Exit 1 is not used (code = 0), control branches to block DE21.

Block DE18: The exit routine at $SWBX executes an SBR which sets the return address, block DE21, into the re-entry routine (at $STLEXT). Index register 15 is restored for the user, and control branches (as set by block DE16) to the Exit 1 address.

Block DE19: Exit 1 is used for entering additional information into the standard output trailer label.

Block DE20: The re-entry routine is executed (at $REENT), to save index register 15 for the user, to reset it for IOCS (from $DTFBX), and to branch to block DE21 (as set by block DE18).

Block DE21: The trailer label is written on the output tape by the subroutine at $WRITRU.

Block DE22, $SUSXA: The Exit 2 check indicator (indexed label, $D2) is tested. If Exit 2 is not used (code = 0), control branches to block DE28.

Block DE23: The user's Exit 2 address in the DTF (indexed label, $E2) is moved to the exit routine (at $SWBXA).

Block DE24: The exit routine at $SWBX executes an SBR which sets the return address, block DE27, into the re-entry routine (at $STLEXT). Index register 15 is restored for the user, and control branches (as set by block DE23) to the Exit 2 address.

Block DE25: Exit 2 is used for writing non-standard labels in addition to, or in lieu of, standard labels.

Block DE26: The re-entry routine (at $REENT) saves index register 15 for the user, resets it for IOCS (from $DTFBX), and branches to block DE27 (as set by block DE24).

Block DE27: A tape mark is written after the label by the utility subroutine at $WTMRU.

POST TRAILER PROCEDURES

Block DE28, $ENTH, and Blocks DE29, DE30, DE31, and DE32: The rewind indicator for the file (indexed label, $DTFL5) is tested for rewind (code = 1) and rewind unload (code = 2). If either is called for, the appropriate utility subroutine at $RWDRU or $RWURU is executed. The reel sequence number (indexed label, $HRS-1) is then updated by 1.

Block DE33: The check character in the calling sequence work area ($DTFBX-5) is checked for an * to test the kind of operation. If it is an * (EOR or FEORL), control goes to block DE34 to prepare for a new tape reel. Otherwise, it is a CLOSE operation and control branches to $ENTJ, block DB25.

Block DE34: An identifying message, 20120bEORbxxxn, where xxxn is the base tape identification, is set up in the console message area.

Blocks DE35 and DE36: The alternate reel indicator for the file (indexed label, DTFL2) is tested. If the code is 1, source DTF ALTDRIVE was specified and control branches to block DE37. Otherwise, the $HALT subroutine is executed to type the message and a waiting loop is entered to permit the operator to mount the new reel. Control then branches to $ENTA, block DB01, to open the new reel.

Block DE37: The base and alternate tape identifications for the file (indexed labels, $AD and $AD-5, respectively) are swapped.

Blocks DE38 and DE39: After the first position in the console message is decremented by 1, the $NOTE subroutine is executed to type 10120bEORbxxxn where xxxn represents the former base tape identification. Control then branches to $ENTA, block DB01, to open the new reel.

Linkage Routines

COMMON ENTRY FROM OPEN, CLOSE, FEORL, AND RDLIN MACROS

Block DF01, $CLOP: An SBR into the A-address of $EXIT, block DF03, is executed. This sets $EXIT to point to the first check character in the macro calling sequence.

Block DF02: The clear channels subroutine is executed to clear both channels of unchecked I/O operations, and IOCS is set to ignore start channel operations. The contents of index register 15 are saved for the user.

CALLING SEQUENCE AND LINKAGE TABLE ROUTINE

Block DF03, $EXIT: The next (or first) segment of the macro calling sequence, consisting of the check character, file reference address, and next check character, is moved to the calling sequence work area. The work area format is:

Label	*Operation*	*Operand*	*Explanation*
$DTFBX	DCW	@x@	Check character for file
		@nnnnn@	File reference address
		@x@	Check character for next file or terminal character

Each macro has its unique check character as follows:

Macro	*Check Character*	*Presence of 2-bit*
OPEN	C	yes
CLOSE	□ or) (lozenge or right paren)	no
FEORL	* (asterisk)	no
RDLIN	? or $\overset{+}{0}$ (question mark or plus zero)	not used

The terminal check character is always a J.

Blocks DF04 and DF05: The calling sequence pointer (A-address of $EXIT) is set to point to the next file's check character (or terminal character) in the macro calling sequence.

Block DF06: Index register 15 is initialized with the file reference address for the file (from $DTFBX in the calling sequence work area).

Block DF07: The check character for the file (from $DTFBX−5 in the calling sequence work area) is compared to $\overset{+}{0}$ (plus zero). If it is a $\overset{+}{0}$, control branches to process the RDLIN operation at $RDLIN, block DG31.

Block DF08: A look-up high or equal with the file type as the search argument (indexed label, $DTFACT) is made in the routine linkage table. Each table argument (file type) is preceded by the OPEN and CLOSE linkages. For example, the one-area tape linkages are:

Operation	*Operand*	*Explanation*
DCW	$TPCLOS	Close linkage
DC	@bJ@	Needed for the move
DC	$ENTA	Open linkage
DC	1	File type (table arg)

When PRIORITY overlay is specified, the beginning of the table is compiled with an all-inclusive sequence shown below. The 9 in the DC at $POX causes open linkage to the priority assignment routine regardless of file type.

Label	*Operation*	*Operand*
	DCW	$EXITRU
	DC	@bJ@
	DC	$PAHSK
$POX	DC	9

After all two-area files have been processed in the priority assignment routine (the others are passed by), the all-inclusive sequence is overlayed with a tape sequence before the second IOCS load (see Chart DA, block DA28).

Block DF09: After an SBR into the A-address of a MLCA, the latter moves the OPEN and CLOSE linkages into a double branch, which is shown as the exits from block DF10.

Block DF10: The check character for the file (from $DTFBX−5 in the calling sequence work area) is tested for the presence of a 2-bit to determine which branch to take. The presence of a 2-bit causes control to go to the OPEN branch; its absence, to the CLOSE branch. (See table at block DF03.) Chart DF shows a summary of linkages for each branch, OPEN, or CLOSE, by file type. The file types are in the left-most column of the tables.

TERMINATION ROUTINE

Block DF11, $EXITRU: The next check character in the calling sequence work area is compared to the termination character J. If it is not J, there is another file to process and control branches to $EXIT, block DF03.

Block DF12: The $ENTRY routine is reset to permit the resumption of channel operations.

Block DF13: The address of the termination character in the calling sequence pointer at $EXIT is moved to the exit routine at $SWBXA, block DF33.

Block DF14: The exit routine is executed (blocks DF31 through DF33) to restore index register 15 for the user, and control branches (as set by block DF13) to the termination instruction. The table below block DF14 shows the effective branch location taken (the operand of the branch instruction) for an end-of-reel operation.

CLEAR CHANNEL ROUTINE

Block DF21, $STLE or $SCS: An SBR into the re-entry routine (at $STLEXT) is executed to establish the return linkage.

Blocks DF22 and DF23: Channels 1 and 2 are cleared of unchecked I/O operations, to prevent any interrupts during label processing, etc., in the DIOCS routines.

Block DF24: The $ENTRY routine is set to prevent the resumption of normal channel operations.

Block DF25: The re-entry subroutine is executed to save the contents of index register 15 for the user. Control then branches to the location as set by block DF21.

EXIT ROUTINE

Block DF31, $SWBX: An SBR into the re-entry routine at $STLEXT, block DF43, is executed in case IOCS will be re-entered from user coding at one of the 8 standard exits.

Block DF32: Index register 15 is restored for the user with its previous contents saved in $X15HD.

Block DF33, $SWBXA: Control branches to the address as set by the DIOCS routine exited from, or to the termination instruction as set by block DF13.

RE-ENTRY ROUTINE

Block DF41, $REENT: The contents of index register 15 are saved for the user in the hold area, $X15HD.

Block DF42: Index register 15 is initialized with the contents of $DTFBX, in case this is a re-entry from user coding during DIOCS routine processing.

Block DF43, $STLEXT: Return is made to the main IOCS routine as set by the SBR at block DF31 or block DF21.

Description of General I/O Routines, RDLIN

READ OR WRITE LABEL ROUTINE

Block DG01, $READRU: An SBR is executed into the routine's exit, block DG10.

Block DG02: All 80 positions of the IOCS label area are blanked.

Block DG03: The I/O command at $LBOP, block DG06, is set to a read by moving an R into its d-modifier.

Block DG04, $LBIN, and Block DG05: The X-control of the base tape identification (indexed label, $AD-6) is moved to the X-control of the I/O command at $LBOP, and the R or X channel status operation code (indexed label, $AD-5) is moved into the operation codes of the BEX and BA instructions that follow the I/O command.

Block DG06, $LBOP: The I/O command is executed to read the label into the IOCS label area or write the label from the IOCS label area (see Figure 4).

Block DG07: A BEX is executed for the channel to $ERROR, block DG08. A branch is taken only for busy, not ready, or data check indications. The BEX forces an interlock on further processing until the operation has been completed.

Block DG08: The error routine is executed, correcting the error if possible.

Block DG09: A BA to the next sequential instruction is executed to prevent interlock on the next I/O command given.

Block DG10: Control exits to the location as set by block DG01 or DG11.

Block DG11, $WRITRU: An SBR is executed into the routine's exit, block DG10.

Block DG12: The I/O command at $LBOP, block DG06, is set to a write by moving a W into its d-modifier. Control then branches to block DG04.

I/O UTILITY ROUTINE: REWIND, REWIND UNLOAD, AND WRITE TAPE MARK

Blocks DG21, DG22, and DG23: $RWDRU, $RWURU, and $WTMRU represent the entry points to the utility routine for rewind, rewind unload, and write tape mark operations, respectively. In each case, an SBR is executed to set the common exit, block DG30. The SBR is followed by a 1-character DCW, which contains the d-modifier that applies for the operation (R, U, or M).

Block DG24: The address of the DCW is moved to the A-address of the move instruction at block DG26.

Block DG25: The R or X channel status operation code for the file (indexed label, $AD-5) is moved to the operation code for the BA instruction at block DG28.

Block DG26: The character in the DCW is moved to the d-modifier of the I/O utility instruction at block DG27. The X-control of the base tape identification (indexed label, $AD-6) is chain-moved to the X-control of the I/O utility instruction.

Block DG27: The I/O utility command is executed.

Block DG28: A BA on the channel to $ERROR is executed. The BA forces an interlock on further processing until the operation has been completed for a write tape mark operation only.

1. For a rewind or a rewind unload operation, a branch can only occur for busy or not ready.
2. For a write tape mark operation, a branch can only occur for busy, not ready, or data check.
3. The BA also serves to satisfy the interlock requirement for the next I/O command.

Block DG29: The error routine is executed, correcting the error if possible.

Block DG30: Control returns to the user's program at the address set by block DG21, DG22, or DG23.

PROCESS RDLIN

Block DG31: The RDLIN card is read into the IOCS label area.

Block DG32: When the operation has been completed, a BA1 is executed to $ERROR.

Block DG33: An error has occurred. A halt is provided to reload the card, before a re-execution of the I/O command in the error routine.

Block DG34: Card columns 16-20 of the card in the IOCS label area are compared to the identifier, @RDLIN@. A branch unequal is made to block DG36.

Block DG35: A RDLIN card was recognized. A series of chained moves are executed to move card columns 21-50 to the several internal fields that relate to label processing in the file table. Control then branches to the termination routine at $EXITRU, block DF11.

Block DG36: It is not a RDLIN card. An identifying message, 20136bRLNb, is prepared in the console message area.

Block DG37: The $HALT subroutine is executed. The message is typed, followed by a waiting loop to allow the operator to insert the proper RDLIN card. Control then branches to block DG31 to read it.

Description of Message and Wait Loop Routine

The function of this routine is to type a message and to enter a wait loop for an operator reply. After the reply is entered, control is returned to the instruction immediately following the one which originally caused entry to the message and wait loop routine.

Block DH01, $HALT: The contents of the B-address register are stored in the I-address of the branch instruction at $HALTX (block DH08). This sets the routine's exit.

Block DH02: The contents of an area labeled $ERFLD are typed on the console printer. $ERFLD is the area used by IOCS to assemble its messages.

Block DH03: A message, "20183 CI," is moved to $ERFLD. This message is typed if the information entered through the console printer by the operator is invalid, or if the operator cancels during inquiry.

Block DH04: A read console printer instruction is executed. If the INQUIRY REQUEST key has not yet been pressed by the operator, the no transfer indicator is turned on and control goes to block DH05. If, however, it has been pressed, an I is typed to indicate an inquiry operation and the keyboard is unlocked to allow entry of data by the operator. If, during inquiry, the operator presses the INQUIRY CANCEL key, the condition I/O channel status indicator is turned on and control goes to block DH05. After entering data, the operator presses the INQUIRY RELEASE key.

Block DH05: A test is made to determine if the no transfer indicator is on. If it is, it indicates the operator has not yet pressed the INQUIRY REQUEST key and control returns to block DH04 to retry the read console printer instruction.

Block DH06: A test is made to determine if the I/O condition or data check I/O channel status indicators are on. If either is on, it indicates that a validity error was detected on the entry of data or the INQUIRY CANCEL key was operated during inquiry; control passes to block DH02 to type the console entry error message.

Block DH07: The I/O channel status test is satisfied by executing a branch any to the next sequential instruction.

Block DH08, $HALTX: Control branches to the location set by block DH01.

WRITE CONSOLE PRINTER ROUTINE

The function of this routine is to type the contents of $ERFLD on the console printer and to return control to the instruction immediately following the one which caused entry to the write console printer routine.

Block DH11, $NOTE: The contents of the B-address register are stored in the I-address of the branch instruction at block DH15. This sets the routine's exit.

Block DH12: The contents of $ERFLD are typed on the console printer.

Block DH13: A test is made to determine if any I/O channel status indicators are on. If they are all off, control goes to block DH15 to exit from the routine. If any indicators are on, control enters the console printer error routine (represented by block DH14).

Block DH14: The console printer error routine determines if the operation must be retried because of a busy or data check condition. If either condition is present, control returns to block DH12 to retry typing of the message.

Block DH15: Control branches to the location set by block DH11.

CONSOLE PRINTER ERROR ROUTINE

If this routine is entered because of a data check, I/O condition, no transfer, or busy condition, it sends control back to the unsuccessful console printer instruction. If entered for some other channel status indicator, it returns control to the instruction immediately following the one which caused entry to it. Wrong length records are not checked.

Block DH21, $CPERR: The contents of the B-address register are stored in the I-address of the branch at $CPEX. This sets the normal (no error) return.

Block DH22: A test is made to determine if the data check, I/O condition, no transfer, or busy indicators are on. If any are on, the operation must be re-executed; control goes to block DH23. If none are on, the operation is considered a success, and control goes to block DH24.

Block DH23: A + 17 is subtracted from the I-address of the branch at $CPEX. This sets the routine's exit to return to the unsuccessful console printer instruction.

Block DH24, $CPEX: Control branches to the location set by block DH21 or by block DH23.

SAVE ROUTINE

The save routine translates the settings of the zero balance and compare indicators into a code which is saved in core storage. If the program uses any special interrupts, e.g., URREQUEST or INQUIRY, the save routine will appear in the interrupt coding. Otherwise, it appears in the tape error routine. The code generated by the save routine is used by the restore coding, described later, to restore the indicator settings before control is given back to the user.

The results of the save routine for various combinations of the zero balance and compare indicators are shown in the following table. (Note: Units position is labeled $PS.)

	Zero Balance	*No Zero Balance*
	vvv	v v
Low	102	102
	vvv	v v
Equal	101	101
	vvv	v v
High	100	100

Block DH31, $PSV: The contents of the B-address register are stored in the I-address of the branch at $PSX. This sets the routine's exit.

Block DH32: The code, located at $PS, $PS-1, and $PS-2, is set to 101 with word marks over all three characters.

Block DH33: A test is made to determine if the zero balance indicator is on. If it is, control goes to block DH35.

Block DH34: The word mark at $PS-1 is cleared.

Block DH35: A test is made to determine if the equal compare indicator is on. If it is, control goes to block DH39.

Block DH36: A test is made to determine if the low compare indicator is on. If it is, control goes to block DH38.

Block DH37: The units position of the code ($PS) is made equal to zero by subtracting it from itself.

Block DH38: The units position of the code ($PS) is added to itself.

Block DH39, $PSX: Control branches to the location set by block DH31.

RESTORE ROUTINE

The restore coding appears in line in various places in IOCS. It is not a subroutine. The function of the restore coding is to recreate the 1411 machine status, as it existed before the save routine was executed, using the code generated by the save routine.

Block DH41: The units position of the code ($PS) is compared to a 1. This instruction restores the high, low, or equal compare indicator.

Block DH42: The zero balance status is restored by executing a zero and add instruction. The field added is the tens position of the code ($PS-1).

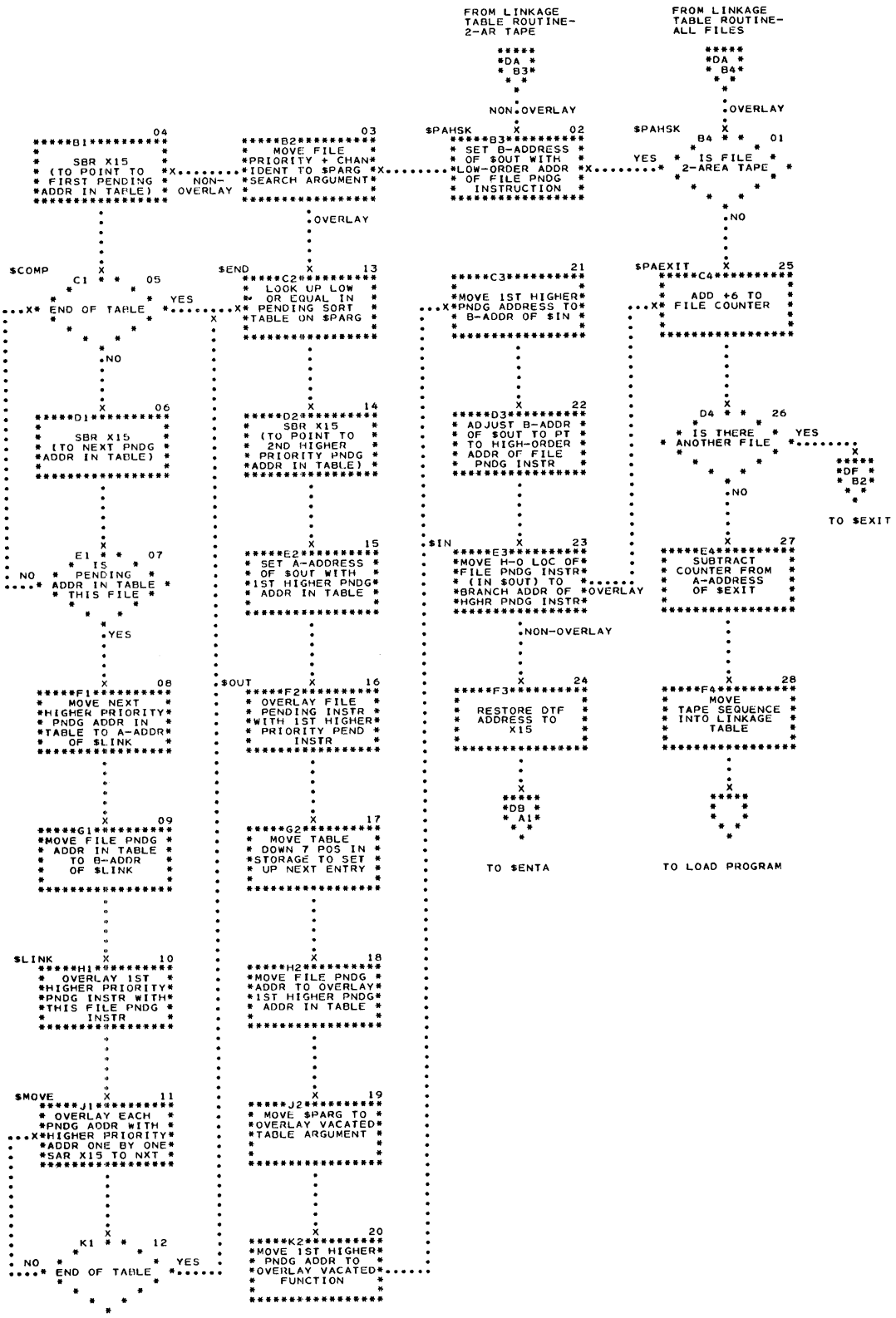


Chart DA. Priority Assignment

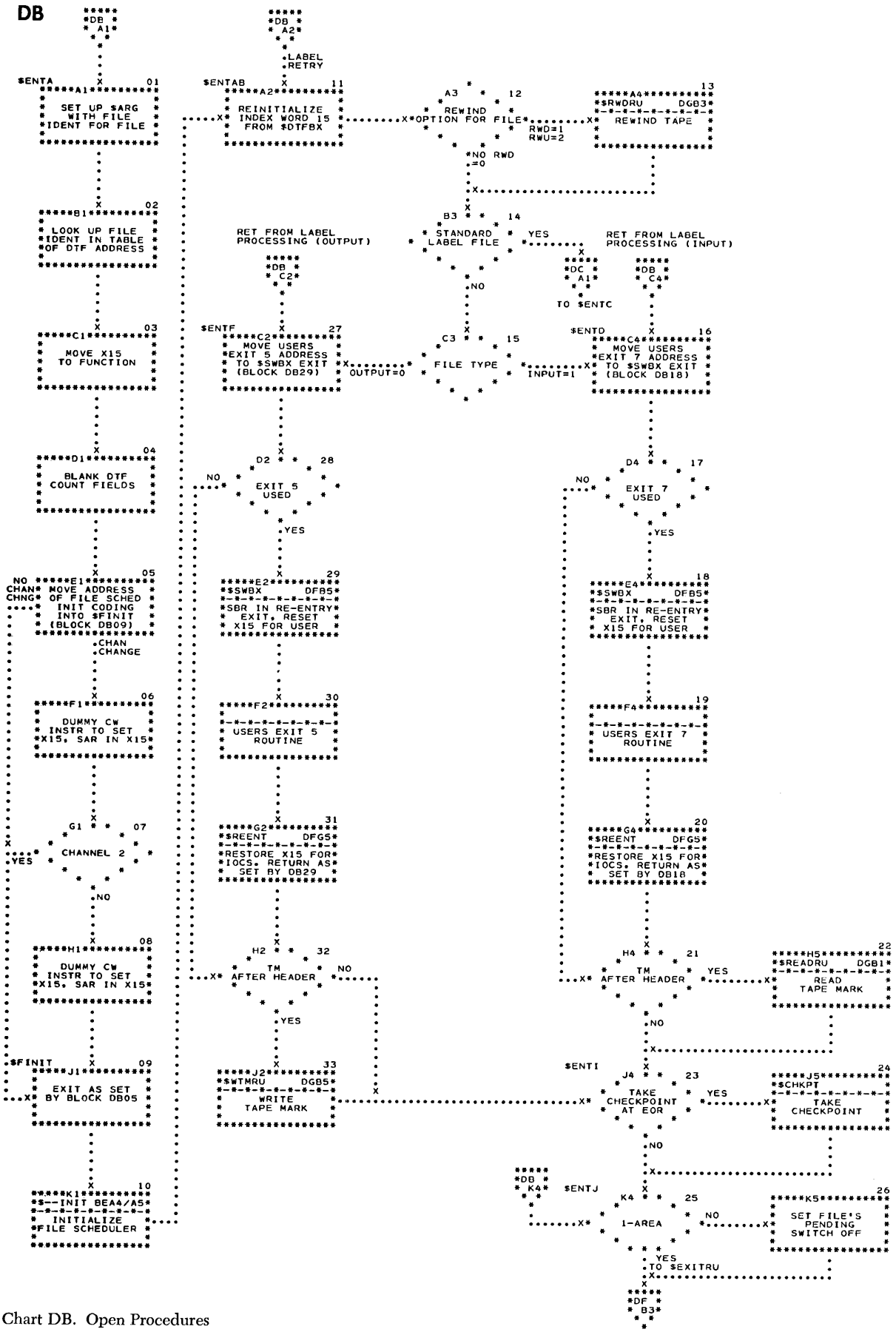
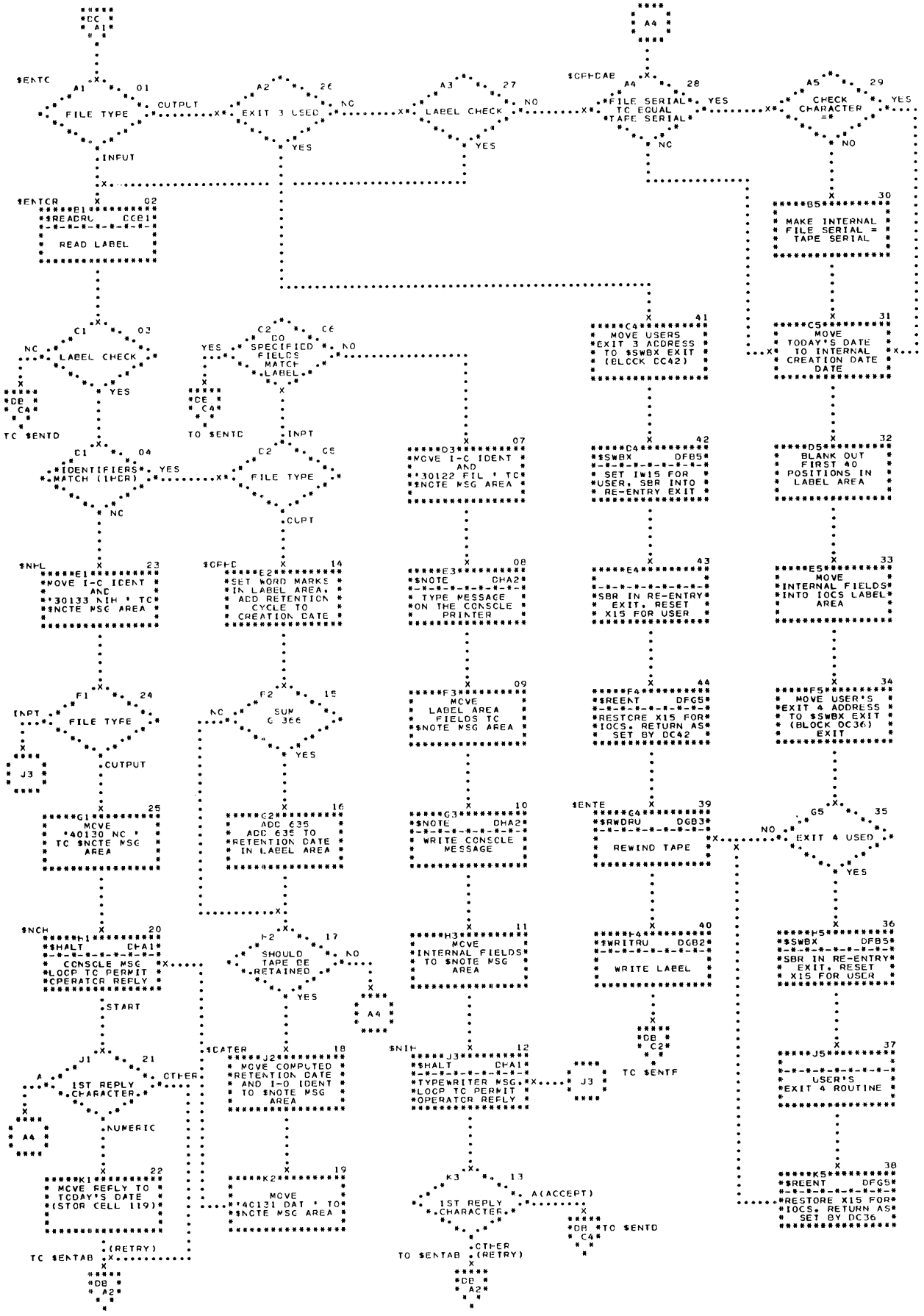


Chart DB. Open Procedures



● Chart DC. Header Label Procedures

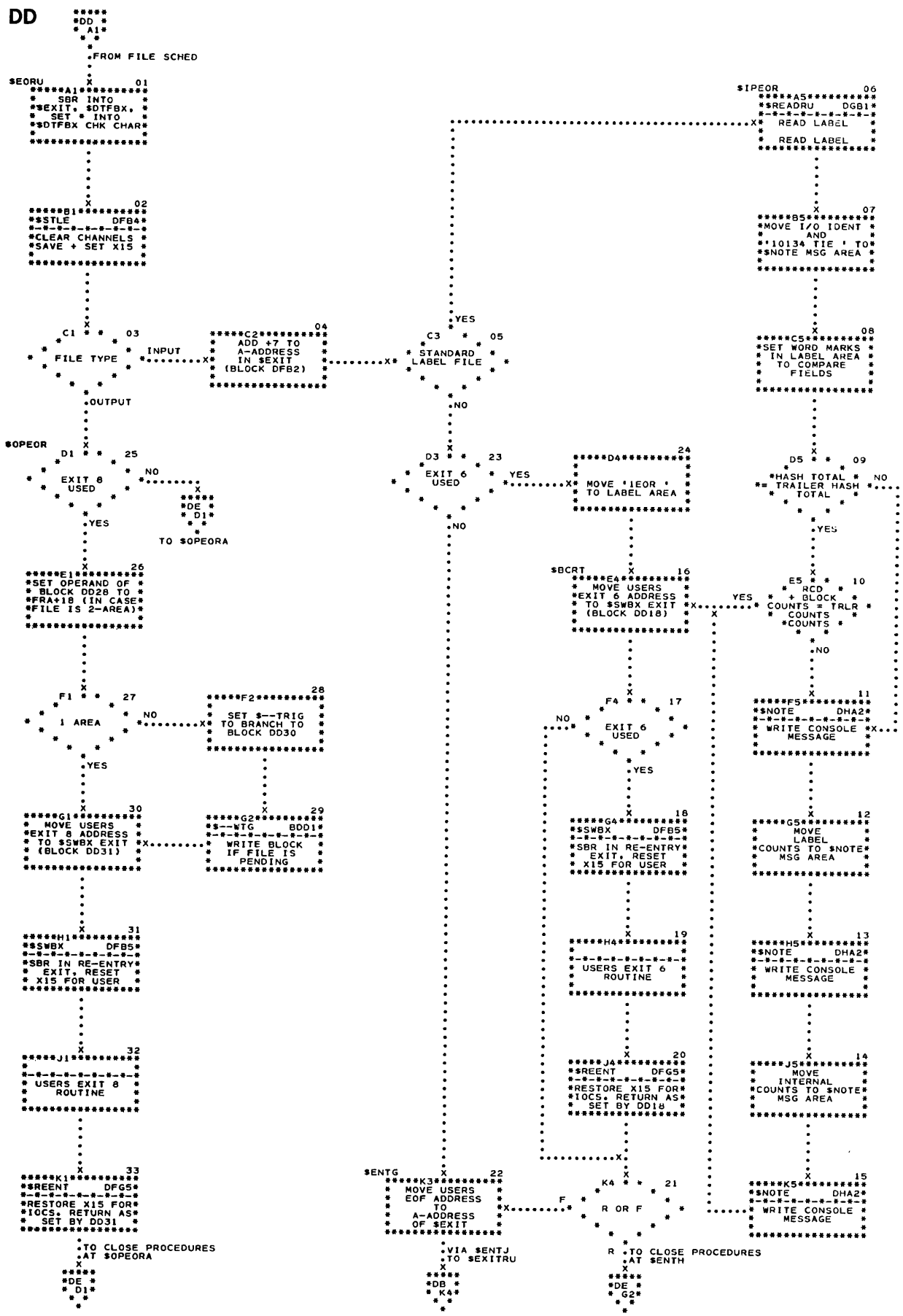
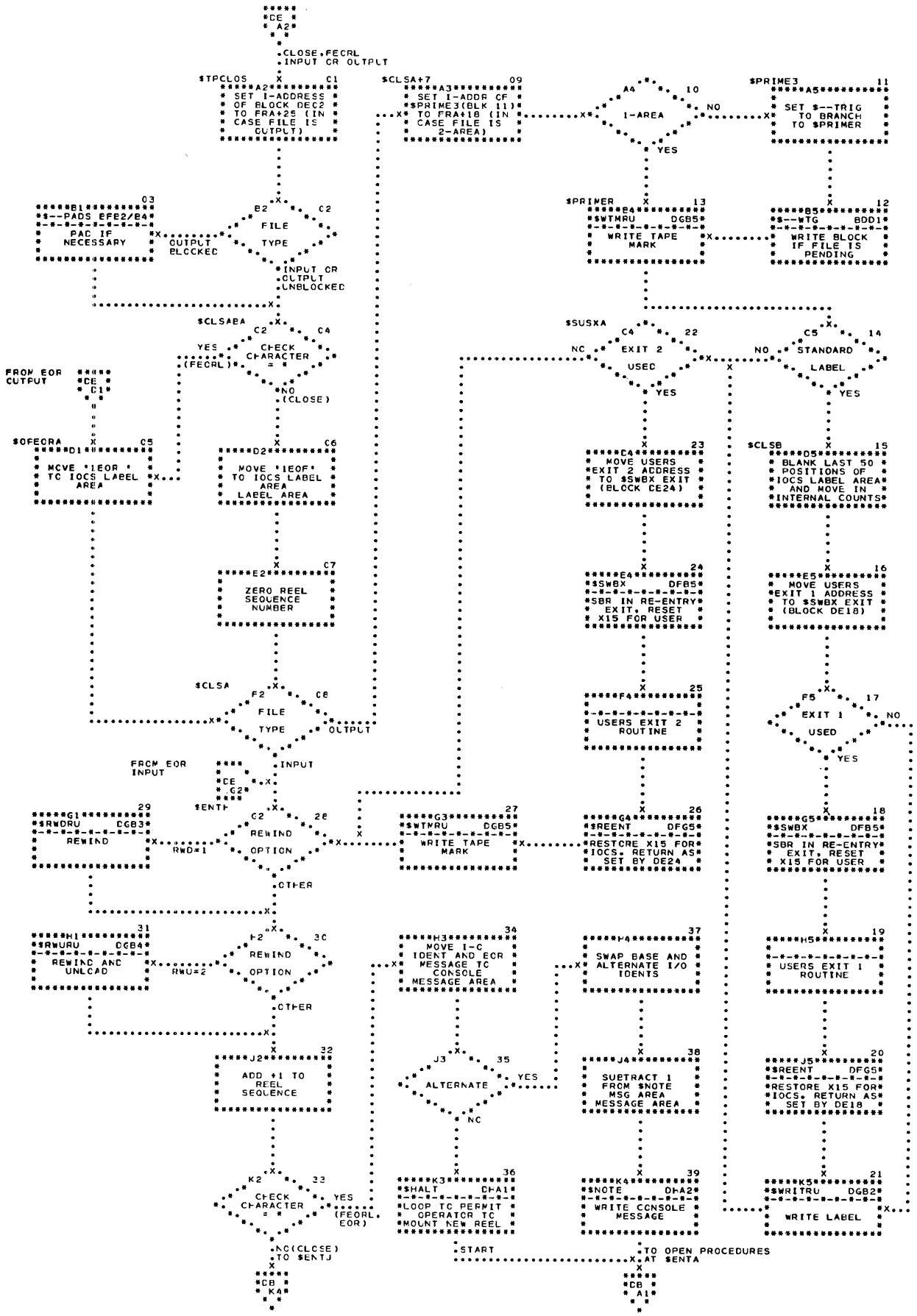
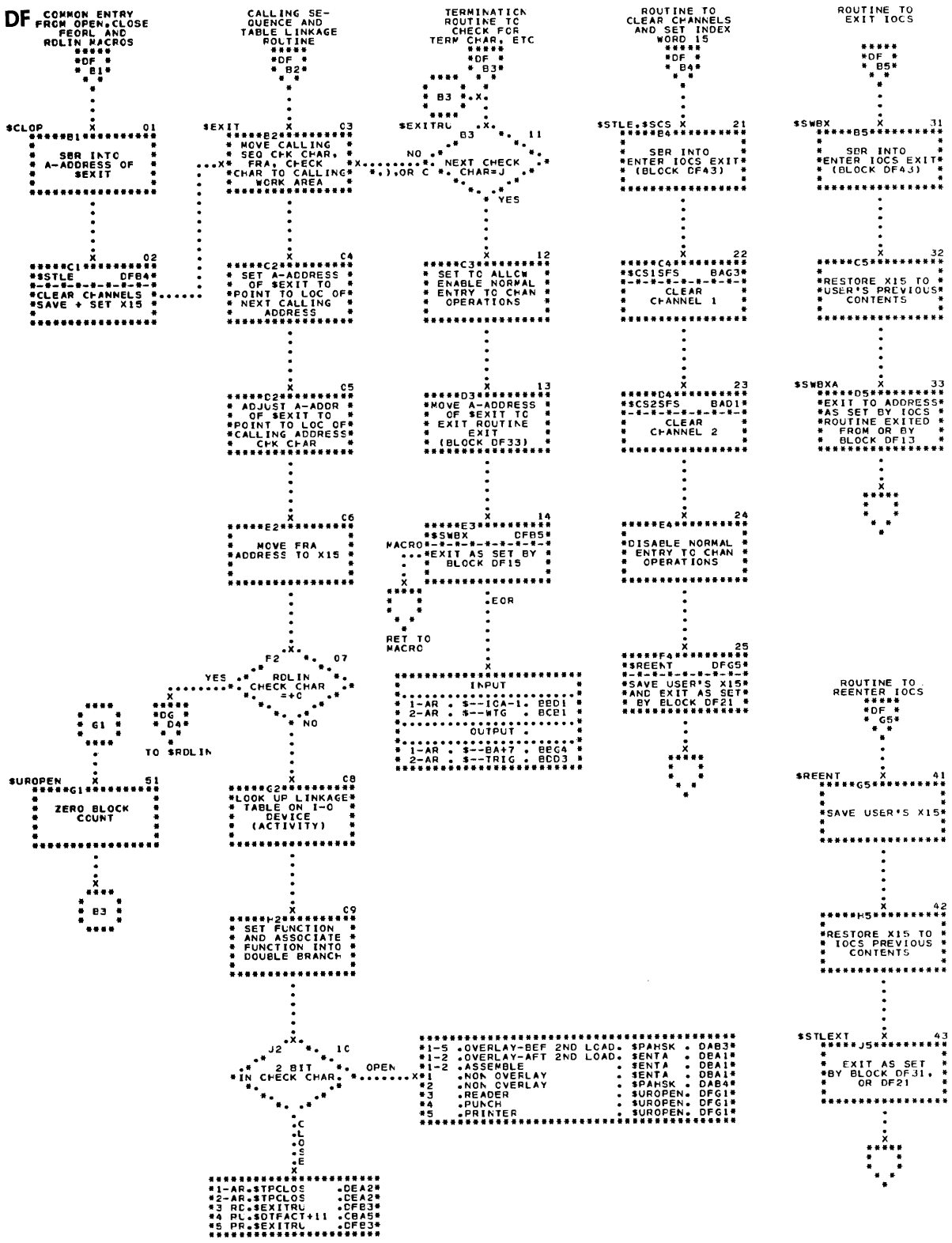


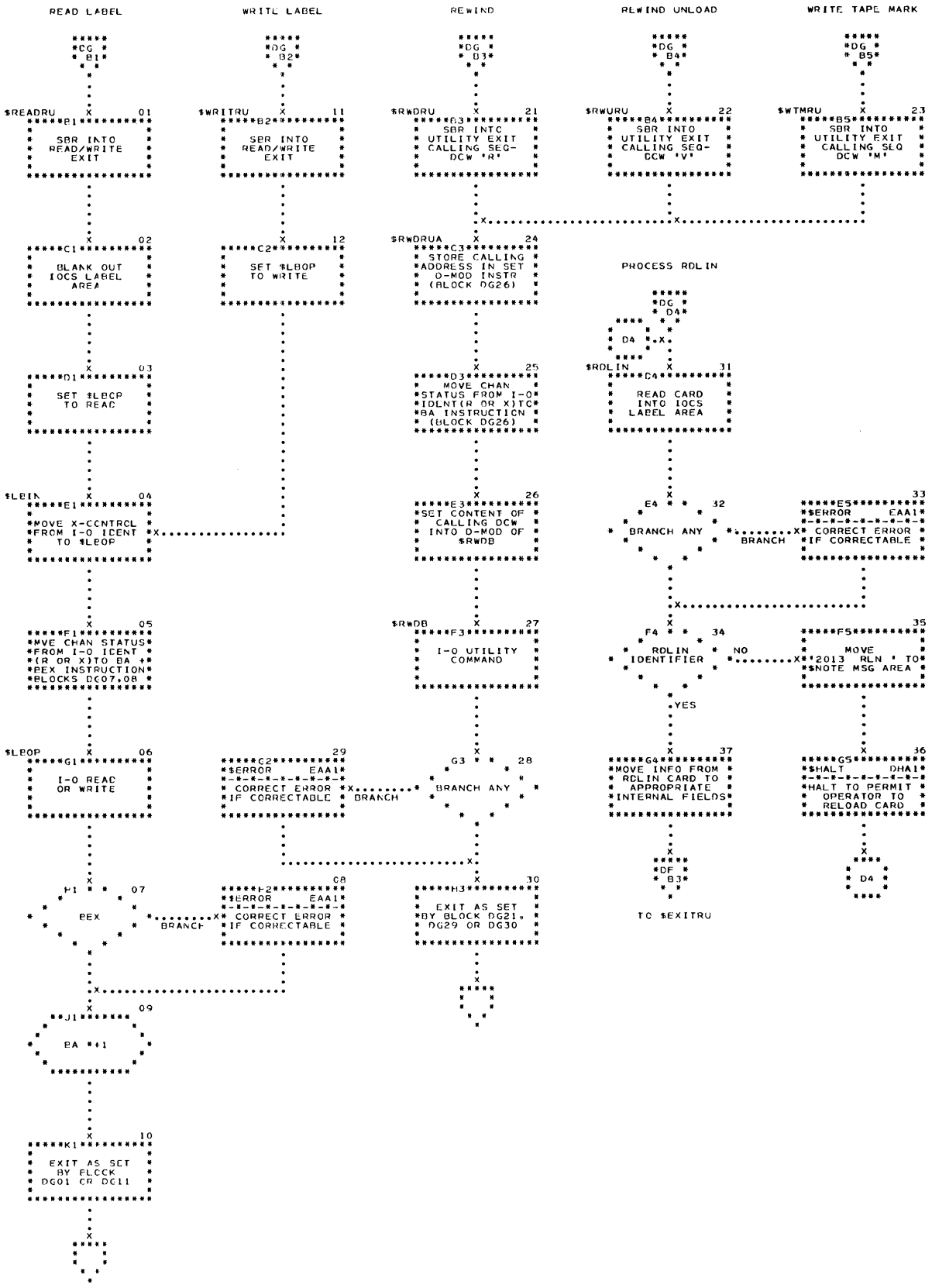
Chart DD. End-of-Reel Procedures



● Chart DE. Close Procedures



- Chart DF. Linkage Routines



- Chart DG. General i/o Routines

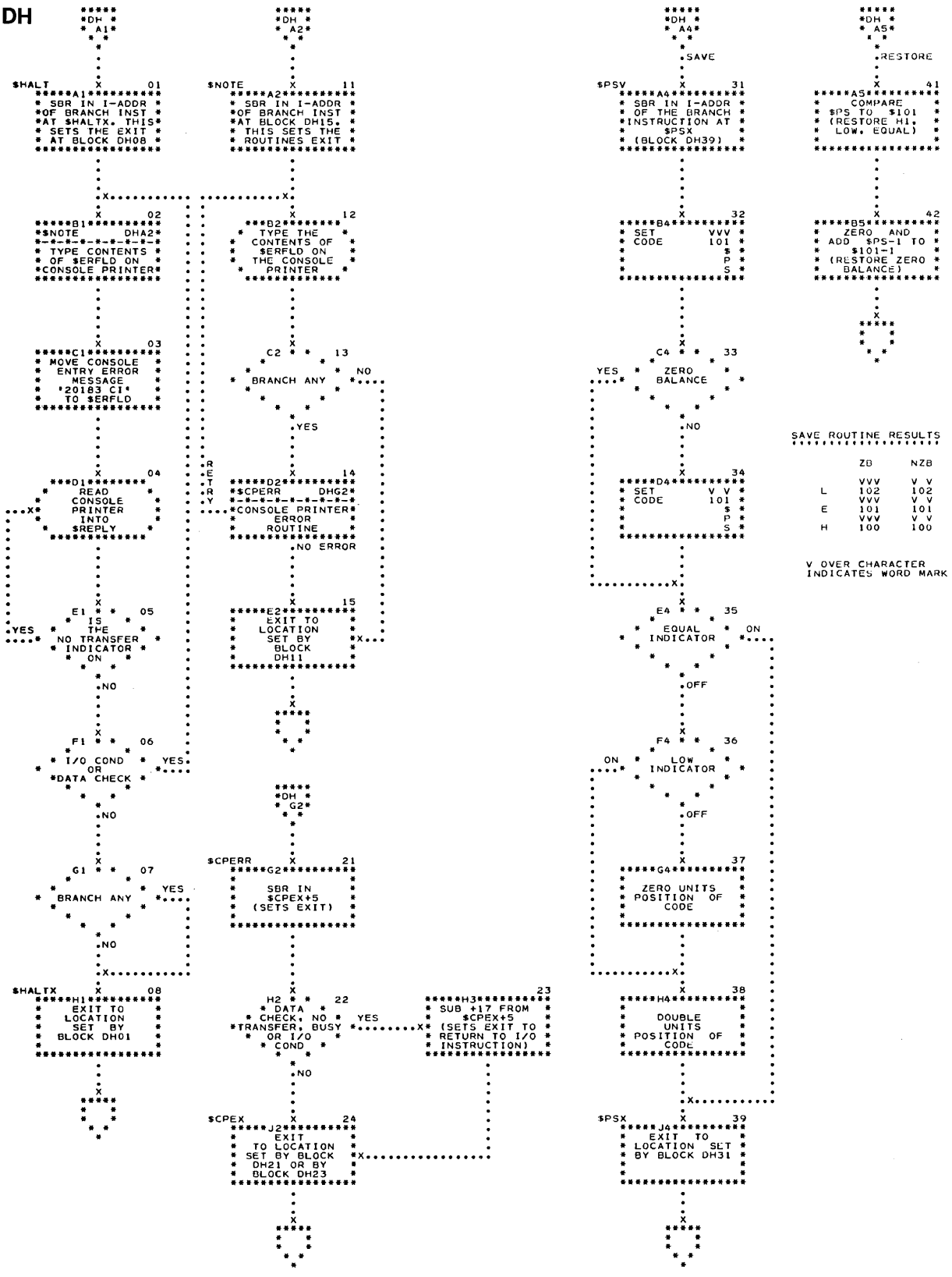


Chart DH. Message, Reply, Save, and Restore

Error Routines

LOOKUP OF FILE REFERENCE ADDRESS

The table of file reference addresses is used by the tape error routine to look up (low or equal) the starting address of a file reference table knowing only the file identification (channel and unit).

The table is in order by unit number (9 to 0) and holds ten or twenty entries, depending on whether there are one or two channels operative. Each entry is a seven-position DCW. The first five positions (function) hold a file reference address or, for an inactive file, blanks. The sixth and seventh position (table argument) identify channel and unit respectively. The format of the table of file reference addresses is shown for two-channel operation in Figure 5.

```
Label     Operation     Operand        Notes
          DCW           bbbbb0¤        Channel 2, unit 0
          DCW           bbbbb0%        Channel 1, unit 0
                        .......
                        .......
          DCW           bbbbb9¤        Channel 2, unit 9
          DCW           bbbbb9%        Channel 1, unit 9
$CU       EQU           *-1
```

Figure 5. Table of File Reference Addresses

The address of the table argument at the start of a table look-up operation is location $CU. The table is searched until a match is made on file identification. The function of the matching entry contains the starting address of the file reference table for the file corresponding to the search argument. Note that the percent sign (%) and lozenge (¤) for channel-mode is used regardless of whether the assembly is overlap or non-overlap; their collating sequence is lower than an at-sign (@) or asterisk (*), respectively.

Tape Error Routine — Part 1

Block EA01, $ERROR: The contents of the B-register are stored in the I-address of the branch instruction at $EREX (normal exit at EB29). The contents of the B-register are decremented by 1 to get the address of the low-order character of the Channel BA or BEX instruction, and are then stored in the A-address of a move instruction labeled $ERENT, block EA02.

Block EA02, $ERENT: A move instruction, initialized by block EA01, is executed. Its execution places the Channel BA or BEX instruction in an area labeled $ERBA (block EA23). After the move is executed, the content of the A-register, which is now the address of the low-order character of the instruction preceding the Channel BA or BEX instruction, is stored in the A-address of a move instruction at $ERPU, block EA04. This initializes a loop which gets the I/O instruction to $JUG (label of the re-execute area). The loop is necessary because of the instruction format of two-area file schedulers.

Block EA03: A word mark is cleared at $JUG. This initializes the instruction length test.

Block EA04, $ERPU: A move instruction is executed. If entered from block EA03, the instruction preceding the Channel BA or BEX instruction is moved to $JUG (re-execute area). If entered from block EA06, it moves the instruction preceding the last one it moved to $JUG (re-execute area). A SAR instruction is executed to re-initialize the move at $ERPU for another pass, if required.

Block EA05: A test is made to determine if the last instruction moved to $JUG is ten characters in length. If it is, control goes to $ER10 (block EA10).

Block EA06: A test is made to determine if the instruction at $JUG is a 5-character instruction. If it is not five characters in length, control goes to block EA04 to move another instruction to $JUG.

Block EA07: A move instruction is executed to left justify the 5-character I/O instruction in the re-execute area. The label ($JUG) now refers to the operation code of the I/O instruction. A NOP word mark is moved to the location immediately following the instruction to allow its proper execution.

Block EA08: The console message field ($ERFLD) is set to 15 characters in length.

Block EA09: The 5-character I/O instruction is moved to the console message field ($ERFLD). Control goes to block EA12.

Block EA10, $ER10: The console message field ($ERFLD) is set to 21 characters in length.

Block EA11: The 10-character I/O instruction is moved to the console message field.

Block EA12: If the interrupt routine stores the 1411 status indicators because of URREQUEST, INQUIRY, 1414 on Channel 1 or 2, or Disk on Channel 1 or 2, control goes to block EA14.

Block EA13: The 1411 status (compare and zero balance indicators) is saved by a subroutine labeled $PSV.

Block EA14: The channel and unit of the tape file being checked are moved to $ARG.

Block EA15: A table look-up instruction is executed. The search argument is $ARG (channel and unit) and the table is labeled $CU. The table and its description are shown in Figure 5. The function found is the file reference address (address of file name label). The contents of the B-register (low-order address of the function) are stored in the A-address of the move instruction executed in block EA19.

Block EA16: The retry (SERCT) and noise record (SERCT-2) counts are zeroed.

Block EA17: The I-address of the branch instruction at SEREX is moved to the B-address of a BCE instruction at SERQLB (block EC45). This initializes a test which determines if this entry to the error routine was from a label record read operation.

Block EA18: A word mark is set at SERNR+13 (block EB11), a 1-time switch used to prevent multiple printouts of the not ready message. A word mark is cleared at SERSK, a 1-time switch which is used to disable the skip and blank instruction at block EC15 on the first pass. The execution of the CW instruction also initializes a SAR instruction. The contents of the A-register are stored in the I-address of the branch at SERBA. This initializes the branch at SERBA (block EA23) to allow entry to block EA19.

Block EA19: The move instruction, initialized by block EA15, is executed. It moves the file reference address to the A-address of a move instruction at SERFA (block EB31). This move initializes the exceptional condition (WLR or EOF condition) segment of the error routine. This segment is labeled SERDLY (block EB30).

Block EA20: The subroutine SERCHOP initializes the BCB at block EA21 in order to check the proper channel for the file under test.

Block EA21: A test is made to determine if the busy I/O channel status indicator is on. If the channel is not busy, control goes to block EA26.

Block EA22, $JUG: An attempt is made to re-execute the I/O instruction.

Block EA23, $ERBA: A test, initialized by blocks EA02 and EA18, is made. This causes processing to stop until the I/O operation is terminated. The test instruction has the same operation-code and d-modifier as the Channel BA or BEX instruction. If any of the indicators tested are on, control returns to block EA19.

Block EA24: The BA instruction at block EA25 is initialized, to check the proper channel for the file under test.

Block EA25: A BA instruction to the next sequential instruction is executed to satisfy the I/O channel status test before another I/O operation. Control goes to SERLV (block EB26).

Block EA26: A test is made to determine on which channel the I/O operation was executed. If it occurred on Channel 1, control goes to block EB01. If it occurred on Channel 2, Channel 1 must be checked and cleared before proceeding so that the error routine may use the console printer.

Block EA27: The address of the instruction following the Channel 2 BA or BEX instruction is stored in the I-address of a branch at SWTGX (block EA35) to initialize the return to finish checking the Channel 2 operation.

Block EA28: A test is made on Channel 1 to determine if an overlapped operation is in progress or has been completed and not checked. If neither condition exists, Channel 1 is clear and control proceeds to block EB01 where further checking of the I/O operation is performed.

Block EA29, $LMWTGR: The I-address of a branch instruction at SWTGX is decremented by 7. This sets SWTGX (block EA35) to return to the Channel 2 BA or BEX instruction.

Block EA30: The I-address of the branch at SCS1SFX is saved. This is necessary as this linkage to the Channel 1 file scheduler may be destroyed if the error routine forces the Channel 1 I/O operation.

Block EA31: If the interrupt routine stores the 1411 status indicators because of URREQUEST, INQUIRY, 1414 on Channel 1 or 2, or DISK on Channel 1 or 2, control goes to block EA33.

Block EA32: The zero balance and compare indicators are restored.

Block EA33: Channel one is cleared of all unchecked I/O operations now in progress or already completed by giving control to SCS1SFS.

Block EA34: The linkage saved in block EA30 is restored.

Block EA35, $WTGX: Control returns to the error routine (SERROR, block EA01) via the Channel 2 BA or BEX instruction. A brief summary of what has occurred follows:

The error routine was entered for a Channel 2 file. Once in the routine, it was determined that Channel 1 must be cleared and checked. Linkage to the Channel 2 file was saved and the Channel 1 operations were cleared and checked. The error routine was re-entered for the Channel 2 file.

Block EA36, $ERCHOP: The address of the operation code of the BEX instruction being initialized is placed in the B-address of a move instruction at EA37. The address is also stored in the I-address of the exit branch instruction at block EA38.

Block EA37: The move instruction, initialized by block EA36, is executed. It moves the operation code of the Channel BA or BEX instruction to the operation code of the BEX instruction being set to test the proper channel.

Block EA38: A branch is executed to the location set by block EA36.

Tape Error Routine — Part 2

Block EB01: The BNR instruction at block EB02 is initialized to test the proper channel for the file being checked.

Block EB02: A test is made to determine if the not ready I/O channel status indicator is on. If the unit is not ready, control goes to $ERNR (block EB10).

Block EB02.1: With a non-overlapped assembly, block EB03 does not exist. Instead, for an input file, the contents of the B-register, after the read operation, are stored in $ERNOIS to initialize the noise length record test. This is accomplished in the file scheduler before checking the I/O operation.

Block EB03: A test is made to determine on which channel the I/O instruction was executed. If it was Channel 1, the contents of the E-register are stored in $ERNOIS. If Channel 2, the contents of the F-register are stored in $ERNOIS. This initializes the noise length record test starting at block EC04.

Block EB04: The BEF instruction at block EB05 is initialized to test the proper channel for the file being checked.

Block EB05: A test is made to determine if the I/O condition channel status indicator is on. If it is, a branch to $EREF (block EB14) is executed.

Block EB06: The BER instruction at block EB07 is initialized to check the proper channel for the file being tested.

Block EB07: A test is made to determine if the data check I/O channel status indicator is on. If it is, a branch to $ERDC (block EC01), is executed.

Block EB08: A test is made to determine if the I/O operation was a read instruction. If it was, a branch to block EB16 is executed.

Block EB09: The zero length record message, "20117 ZLR," is moved to the console message field, $ERFLD. This message indicates that the first character in the core storage area used for a write operation was a group mark/word mark. This condition sets the WLR indicator which caused the entry to the error routine. A branch to $ERHLT (block EB18) is executed.

Block EB10, $ERNR: (Entry from block EB02.) The not-ready message, "10100 NR," is moved to the console message field ($ERFLD).

Block EB11: If a word mark exists at $ERNR+13, a branch is executed to block EB12. If not, control goes to block EB13. The word mark over the branch operation acts as a 1-time switch and prevents multiple print-outs of the not ready message.

Block EB12: The contents of $ERFLD (not-ready message) are typed on the console printer.

Block EB13: The word mark at $ERNR+13 is cleared to prevent typing multiple not ready messages. A branch to $JUG (block EA22) is executed.

Block EB14, $EREF: The address of $EORU (address of end of reel routine) is stored in $ERAD. The contents of $ERAD may be used by the error routine to initialize linkage for exceptional conditions (WLR or EOF).

Block EB15: A test is made to determine if the EOF condition was tested by the Channel BA or BEX instruction. If it was, the error routine must set up linkage for the EOF exceptional condition and control goes to $ERDLY (block EB30). If EOF was not tested, it indicates the main program treats this condition and control goes to $ERLV (block EB26).

Block EB16: A test is made to determine if the read operation, which is generating WLR checks, has been retried ten times. If it has not, control goes to $ERDC for another attempt at correcting it. If it has been retried ten times, it is considered by IOCS to be uncorrectable and control goes to block EB17 to initialize the exceptional condition linkage for the WLR condition.

Block EB17: The file reference address (address of the file name label) is moved to $ERAD. A +33 is subtracted from the contents of $ERAD to set up linkage to the user's wrong-length-record routine. Control goes to $ERDLY (block EB30).

Block EB18, $ERHLT: The contents of $ERFLD (console message) are typed out on the console printer. The message can indicate a label read error, zero length record, or data check on read. A wait loop is entered, allowing the operator to select an option and enter it through the console printer. After the option is entered, control passes to block EB20.

Block EB20: The first character of the option entry (content of the high-order location of $REPLY) is moved to the d-modifier position of a BCE instruction. This initializes the option test instruction.

Block EB21: The option test instruction is a BCE which compares the contents of a location in the $EROPTN field to the d-modifier set by block EB20.

The contents of the $EROPTN field (six characters) are modified by the DIOCS read error entry in the following manner:

DIOCS READERROR *Entry*	$EROPTN *Field*
No entry — no * scan or dump routines generated	@bbSRPb@
SCAN — no dump routine generated	@b*SRPb@
TAPE, CU — no * scan routine generated	@DbSRPD@
SCAN, TAPE, CU — both generated	@D*SRPb@

The character at $EROPTN−1 is compared to P. If this d-modifier is a P, IOCS ignores the error and control goes to $ERLV (block EB26) to exit the error routine.

Block EB22: If the d-modifier is an R, IOCS will again attempt to execute the I/O operation and control goes to $ERDC (block EC01).

Block EB23: If the d-modifier is an S, IOCS will ignore the I/O operation in error and will read in the next record or block of records. Control goes to $JUG (block EA22).

Block EB24: If the d-modifier is an asterisk and SCAN has been specified in the DIOCS READERROR entry, IOCS

will type out the location(s) of the asterisk(s) in the record. Control goes to $ERSCN (block EC31).

Block EB25: If the d-modifier is a D and a dump tape has been specified in the DIOCS READERROR entry, IOCS will write the record in error on the specified dump tape. Control then goes to $ERDMP-19 (block EC23). If none of the option branches are taken, control goes to $ERHLT (block EB18) to notify the operator that another option must be selected.

Blocks EB26, $ERLV and EB27: If the error routine saves the 1411 status indicators they are restored before exiting from the routine. A branch to $EREX (block EB29) is executed.

Block EB29, $EREX: The I-address of the branch at $EREX is normally that address which had been set at block EA01($ERROR). If, however, an exceptional condition (WLR or EOF) exists for a one-area tape file, the I-address is the file reference address (address of file name) minus seven. The branch is executed.

Block EB30, $ERDLY: The I-address of the branch at $EREX is moved to the I-address of the branch instruction at $EREX + 7. The A-address of a move instruction labeled $ERFA is incremented by 18.

Block EB31, $ERFA: A move instruction, initialized by blocks EA19 and EB30, moves the contents of the file reference address + 18 (address of $EREX + 5 for a one-area file or the address of $--TRIG+5 for a two-area file) to the C-address of a SBR instruction executed in block EB33.

Block EB32: The contents of $ERAD (address of $EORU for EOF condition or the file reference address −33 for WLR condition) are moved to the I-address of a branch located at file reference address −7.

Block EB33: The file reference address −7 is stored in the address set by block EB31.

Block EB34: The address of the NOP, preceding a one-area tape file I/O instruction, is moved to the I-address of a branch instruction located at $EREX + 14.

Block EB35: A test is made to determine if the I/O instruction was a read operation. If it was, control goes to $ERLV (block EB26).

Block EB36: The I/O channel status test instruction, executed at block EB37, is initialized to check channel.

Block EB37: This block is entered if the I/O instruction was a write operation and the EOF I/O condition exists. A test is made to determine if the data check I/O channel status indicator is on. If it is, the record just written on tape is in error and control goes to $ERDC (block EC01) to attempt to correct it. If it is not, control goes to the error routine exit coding at $ERLV (block EB26).

Tape Error Routine — Part 3

Block EC01, $ERDC: A test is made to determine if the I/O instruction is a write tape mark operation. If it is, the noise record sequence starting at block EC02 is bypassed and control goes to $ERH+7 (block EC08) to initialize for a re-execute attempt or to type a console message and enter a wait loop for operator action.

Block EC02: The 5-character starting address of the I/O area, specified by the I/O instruction, is placed in $ERFLD+14 through $ERFLD+18. This operation initializes the asterisk scan routine.

Block EC03: The starting address of the I/O area is placed in the B-address of a write instruction at $ERDA (block EC27). This initializes the dump routine to write the record, or block of records, in error, on the specified dump tape (specified by DIOCS READERROR ENTRY).

Block EC04: The starting address of the I/O area is placed in $ERBL. This initializes the noise length record test.

Block EC05: The contents of $ERNOIS (contents of E- or F-register as set by block EB03) are subtracted from the contents of $ERBL (the starting address of the I/O area). The contents of $ERBL, after the subtract instruction is executed, is a negative number equal to the number of characters read into core plus one.

Block EC06: This block exists only for a non-overlapped IOCS assembly. The noise record test is bypassed on a non-overlapped write tape operation as $ERNOIS is not initialized for a noise record test.

Block EC07: The contents of $ERBL are compared to −13. If the number of characters read is 13 or more, the high compare indicator is set. A branch high is executed; if the branch is taken, IOCS does not consider the record a noise record and control goes to $ERH+7 (block EC08). If the number of characters read is 12 or less, IOCS considers the record a noise record and control passes to block EC08.

Block EC08, $ERH+7: The contents of $ERBL are incremented by 1, making it equal to the number of characters read into core. The record length (contents of $ERBL) is included in the typing of read data check messages. The retry counter ($ERCT) is incremented by 1 to accumulate the number of executions of the I/O instruction.

Block EC09: A test is made to determine if the retry counter is equal to 20. If it is, it indicates that the I/O operation was executed 20 times in an attempt to correct the error but the failure still exists, and control goes to $ERCTL (block EC38).

Block EC10: The X-control field of the I/O instruction is moved to the X-control field of the skip and blank instruction at $ERSK (block EC15). The X-control field of the I/O instruction is moved to the X-control field of the backspace instruction (block EC11). This initializes the instructions for execution on the proper channel and unit.

Block EC11: The backspace instruction (initialized by block EC10) is executed.

Block EC12: The branch any instruction, executed in block EC13, is initialized to check the proper channel.

Block EC13: A branch any instruction is executed. It is primarily to satisfy the I/O channel status test requirements since, at this point in the error routine, no I/O channel status indicators should be on. If the branch is taken, an attempt is made to re-execute the backspace and control passes to block EC11.

Block EC14: If a word mark does not exist at $ERSK (block EC15), the skip and blank instruction is bypassed and control goes to block EC16.

Block EC15, $ERSK: A skip and blank instruction, initialized by block EC10, is executed.

Block EC16, $ERCHOP: The branch any instruction at block EC17 is initialized to check the proper channel.

Block EC17: A branch any instruction is executed. If the branch is taken, control returns to $ERSK (block EC15) to re-execute the skip instruction. The instruction satisfies the I/O channel status test.

Block EC18: A word mark is set at $ERSK (block EC15). This sets the one-time switch (block EC14) off to allow the execution of the skip instruction on additional re-execute passes. Control goes to $JUG (block EA22).

Block EC19: The noise record count, $ERCT−2, is incremented by 1 to accumulate the number of consecutive noise records.

Block EC20: A test is made to determine if ten consecutive noise length records have been read. If not, control goes to $JUG (block EA22) to re-execute the I/O instruction.

Block EC21: The noise record counter, $ERCT−2, is set to zero.

Block EC22: A noise length record message is moved to the console message area, $ERFLD. Control goes to $ERH (block EC41).

Block 23, $ERDMP−19: This block is entered from block EB25 (DUMP option). The data check message, "60113 DCK," is moved to the console message field, $ERFLD. The message indicates that the record in error was read 20 times without success. A waiting loop was entered after typing a read data check message, and the operator specified the dump option. Control goes to block EC25.

Block EC24, $ERDMP: This block is entered only if the operand of the DIOCS READERROR entry is TAPE,CU. The contents of the console message field, $ERFLD, are changed from "60113 DCK" to "10113 DCK." The new message indicates the writing of the error record on the dump tape is the result of the DIOCS specifications and not because of operator action.

Block EC25: A write tape instruction is executed on the channel and unit specified by the DIOCS READERROR entry. The data written are the contents of the console message field, $ERFLD.

Block EC26: A branch any is executed to the next sequential instruction. This satisfies the I/O channel status test requirements.

Block EC27, $ERDA: A write tape instruction is executed on the channel and unit specified by the DIOCS READERROR entry. The record in error is written on the dump tape.

Block EC28: A branch any is executed to the next sequential instruction. This satisfies the I/O channel status test requirements.

Block EC29: A test is made to determine if the character at $EROPTN is a blank. If it is, it indicates that the dump routine was entered via the option test sequence; IOCS will retype the initial error message and allow the operator to select another option. Control, in this case, goes to $ERCTL (block EC38). If it is not a blank, it indicates that IOCS has automatically written the record in error on the DIOCS specified dump tape and control passes to block EC30 to type out an auto-dump data check message.

Block EC30: The contents of the console message field (auto-dump message) are typed out. Control goes to $JUG (block EA22) to read the next record.

Block EC31, $ERSCN: This sequence of coding does not exist if the DIOCS READERROR entry does not specify SCAN. It is entered from the option test sequence (block EB24) if the operator selects the *SCAN option.

Word marks are set in $ERFLD to facilitate the move instructions used in the routine.

Block EC32: A move instruction is executed which places the starting address of the I/O area (contents of $ERFLD+14 through $ERFLD+18) in the B-address of a BCE instruction at block EC33. The starting address of the I/O area was placed in $ERFLD+14 through $ERFLD+18 by a move instruction at block EC02.

Block EC33: A test is made to determine if the location specified by the B-address of the BCE instruction (character under test) contains an asterisk. If it does, control goes to block EC34. If it does not, control goes to block EC36.

Block EC34, $ERPA: The console message field is set to five characters in length. The location of the asterisk is moved to the console message.

Block EC35: The location of the asterisk is typed by the console printer.

Block EC36: The B-address of the BCE instruction is incremented by +1. This initializes the BCE to test the next location. The record length field ($ERFLD+25) is decremented by one. The record length was placed

in $ERFLD+21 through $ERFLD+25 by a move instruction at block EC43.

Block EC37: A test is made to determine if the subtract instruction executed in block EC36 turned on the zero balance indicator. If it did, it indicates that all characters were tested and control passes to $ERCTL (block EC38). If the zero balance indicator is not on, control returns to block EC33 to test another character.

Block EC38, $ERCTL: The retry counter ($ERCT) is set to zero.

Block EC39: A test is made to determine if the I/O operation was a read instruction. If it was, control goes to block EC43.

Block EC40: "20114 DCK" is moved to the console message field, $ERFLD. This message indicates a data check on a write tape or write tape mark operation. IOCS first backspaced the tape and attempted to rewrite it but the error persisted. A backspace-skip-rewrite sequence was executed 18 times but the record could not be written successfully.

Block EC41, $ERH: A branch is executed to $HALT. The message in the console message field, $ERFLD, is typed and a wait loop is entered to allow operator action. The only option available is to retry the operation. When the operator selects the option, control goes to $ERH+7 (block EC08).

Block EC43: The contents of $ERBL (record length) are moved to the console message ($ERFLD+25). This is initialization for the asterisk scan routine. The record length is included in the typing of the read data check message.

Block EC44: "40119 LRE" is moved to the console message, $ERFLD. This message indicates that the error occurred while reading a label.

Block EC45, $ERQLB: A test is made to determine if the I/O operation in error was a label read. If it was, control goes to $ERHLT (block EB18) to type the label read message.

Block EC46: "60113 DCK" is moved to the console message field, $ERFLD. The message indicates a tape read error. IOCS has attempted to read the record successfully 20 times but the error persists.

Block EC47: A test is made to determine if IOCS is to automatically write the error record on the DIOCS-specified dump tape. If it is, control goes to $ERDMP. Otherwise, control goes to $ERHLT (block EB18) where the contents of the console message field, $ERFLD, are typed and a wait loop is entered, allowing operator intervention.

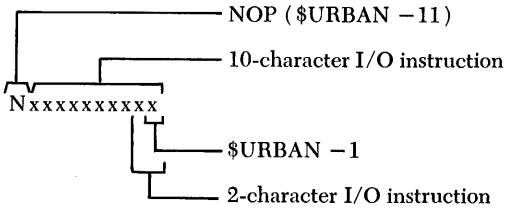
Unit Record Error Routine

Block ED01, $URERR: The contents of the B-register are stored in the I-address of a branch instruction at $UREXIT (block ED34). This initializes the routine's exit. The contents of the B-register are stored in the B-address of a compare instruction at $URQE (block ED27). This initializes a test made in the reader EOF sequence of the error routine.

Block ED02: The contents of the B-register are decremented by +7 to obtain the op-code address of the Channel BA or BEX instruction (the instruction which branched to the error routine). This address is stored in the A-address of a move instruction at $URPKUP (block ED05).

Block ED03, $URENT: A message switch is set on by setting a word mark at $URNR+13 (op-code address of a branch to $NOTE at block ED29). The switch is used to prevent multiple print-outs of the not ready message.

Block ED04: An area is set up for the re-execution of the I/O instruction. The instruction may be of a 2- or 10-character format.



Block ED05, $URPKUP: The move instruction, initialized in block ED02, is executed. It moves the operation code of the Channel BA or BEX instruction to the operation code location of the BA instruction at $URBAN (block ED33) to check the proper channel. The I/O instruction is moved right-to-left to the re-execute area at $URBAN−11 (block ED32).

Block ED06, $URANY: The BCB instruction at block ED07 is initialized to check the proper channel.

Block ED07: A test is made to determine if the unit was busy at the time the I/O instruction was attempted. If it was, control goes to $URBAN−11 (block ED32) to re-execute the I/O instruction.

Block ED08: The BEX instruction at block ED09 is initialized to check the proper channel.

Block ED09: A test is made to determine if any I/O channel status indicator (excluding WLR) is on. If an indicator is on, the load mode test at block ED10 is bypassed and control goes to block ED11 via a BXPA instruction.

Block ED10: A test is made to determine if the I/O instruction is a 10-character instruction executed in load mode. If it is, control goes to $UREXIT (block ED34) to exit from the routine since wrong length records occurring in load mode have little significance.

Block ED11: Channel 1 is cleared of the I/O operation and control passes to block ED12.

Block ED12: A wrong length record message, "20114 WLR," and the I/O instruction are moved to the console message field, $ERFLD.

Blocks ED13, ED14: A test is made to determine if the I/O instruction is 10 characters in length. If it is not, it must be a 2-character instruction and the message in the console message field, $ERFLD is shifted 8 places to the left so that the message is left justified in $ERFLD.

Block ED15: The BWL instruction at block ED16 is initialized to check the proper channel.

Block ED16: A test is made to determine if the wrong length record I/O channel status indicator is on. If it is, control goes to $URTY (block ED35) where a wrong length record message is typed out and a wait loop is entered to enable operator action.

Block ED17: A data check message, "20116 DCK," is moved to the console message field, $ERFLD.

Block ED18: The BER instruction at block ED19 is initialized to check the proper channel.

Block ED19: A test is made to determine if the data check I/O channel status indicator is on. If it is, control goes to $URDCK (block ED37).

Block ED20: A message, "20143 STK," indicating a programming error, is moved to the console message field, $ERFLD.

Block ED21: The BNT instruction at block ED22 is initialized to check the proper channel.

Block ED22: A test is made to determine if the no transfer I/O channel status indicator is on. If it is, it indicates a programming error in the object program, and control goes to $URTY to type the program error message and enter a wait loop for operator action.

Block ED23: The BNR instruction at ED24 is initialized to check the proper channel.

Block ED24: A test is made to determine if the not-ready I/O channel status indicator is on. If it is, the unit is not ready and control goes to $URNR (block ED28). If it is ready, the I/O channel status indicator which caused this entry to the error routine is the I/O condition indicator since at this point all others are off.

Block ED25: A message, "20115 LLC," indicating that the last line printed or the last card punched contained an error, is moved to the console message field, $ERFLD.

Block ED26: A test is made to determine if the I/O instruction is a 10-character print or punch instruction. If it is, control goes to $URTY (block ED35) to type the message assembled in block ED25 and enter a wait loop for operator action.

Block ED27, $URQE: A test is made to determine if the Channel BA or BEX instruction is followed sequentially by a BEX instruction on the same channel. If it is, IOCS assumes that the user checks for EOF on the reader and control goes to $UREXIT (block ED34) to exit the routine.

Block ED28, $URNR: This block is entered if the unit is not ready or if the reader is at EOF and the user does not test it. A not-ready message is moved to the console message field, $ERFLD.

Block ED29, $URNR+13: If a word mark is in location $URNR+13, it indicates that this is the first attempt to type-out the not ready message and control passes to block ED30. If the word mark does not exist, it indicates that the message has been typed once for this condition and the write console printer routine is bypassed to prevent multiple type-outs of the not ready message. Control goes to block ED31.

Block ED30: The not ready message, "10100 NR," is typed by the console printer.

Block ED31: The word mark at $URNR+13 is cleared and control goes to $URBAN−11 to re-attempt the execution of the I/O instruction. A loop in the error routine exists until the device is made ready, at which time the I/O instruction is executed.

Block ED32, $URBAN−11: An attempt is made to execute the I/O instruction.

Block ED33, $URBAN: A test is made to determine if any I/O channel status indicators are on. If none are on, the operation has been performed successfully and control passes to $UREXIT (block ED34) to exit from the routine. If any are on, control returns to $URANY (block ED06) to determine which I/O condition exists and take appropriate action.

Block ED34, $UREXIT: A branch is executed to the location set by block ED01.

Block ED35, $URTY: The message switch is set on (sw at $URNR+13) so that a not ready message may be printed, if that condition exists, after the operator returns control to the error routine.

Block ED36: The contents of the console message field, $ERFLD, are typed out. A wait loop is entered until the operator requests the option of a re-execution of the I/O instruction. He does this by first pressing the INQUIRY REQUEST key and then the INQUIRY RELEASE key. No code word is necessary. Control goes to $URBAN−11 (block ED32).

Block ED37, $URDCK: This block is entered after it is determined that the data check I/O channel status indicator is on. If the data check occurred on a card read operation, control returns to $URTY (block ED35) to type a data check message and enter a wait loop for operator action.

Blocks ED38 and ED39: A +1 is added to a counter and a test is made to determine if the count is odd or even. If odd, it indicates that an attempt has not been made to correct the data check and control goes to $URBAN−11 (block ED32) to re-execute the instruction. If even, it indicates the operation has been tried twice and is still in error and control goes to $URTY (block

ED35) to type a data check message and enter a wait loop for operator action.

Block ED40, $URCHOP: The contents of the B-register are stored in the B-address of the move instruction executed in block ED42.

Block ED41: The contents of the B-register are stored in the exit.

Block ED42: The operation code of the test instruction at $URBAN (block ED33) is moved to the operation code position of the instruction indicated by the B-register.

Block ED43: An exit is made to the location set by block ED40.

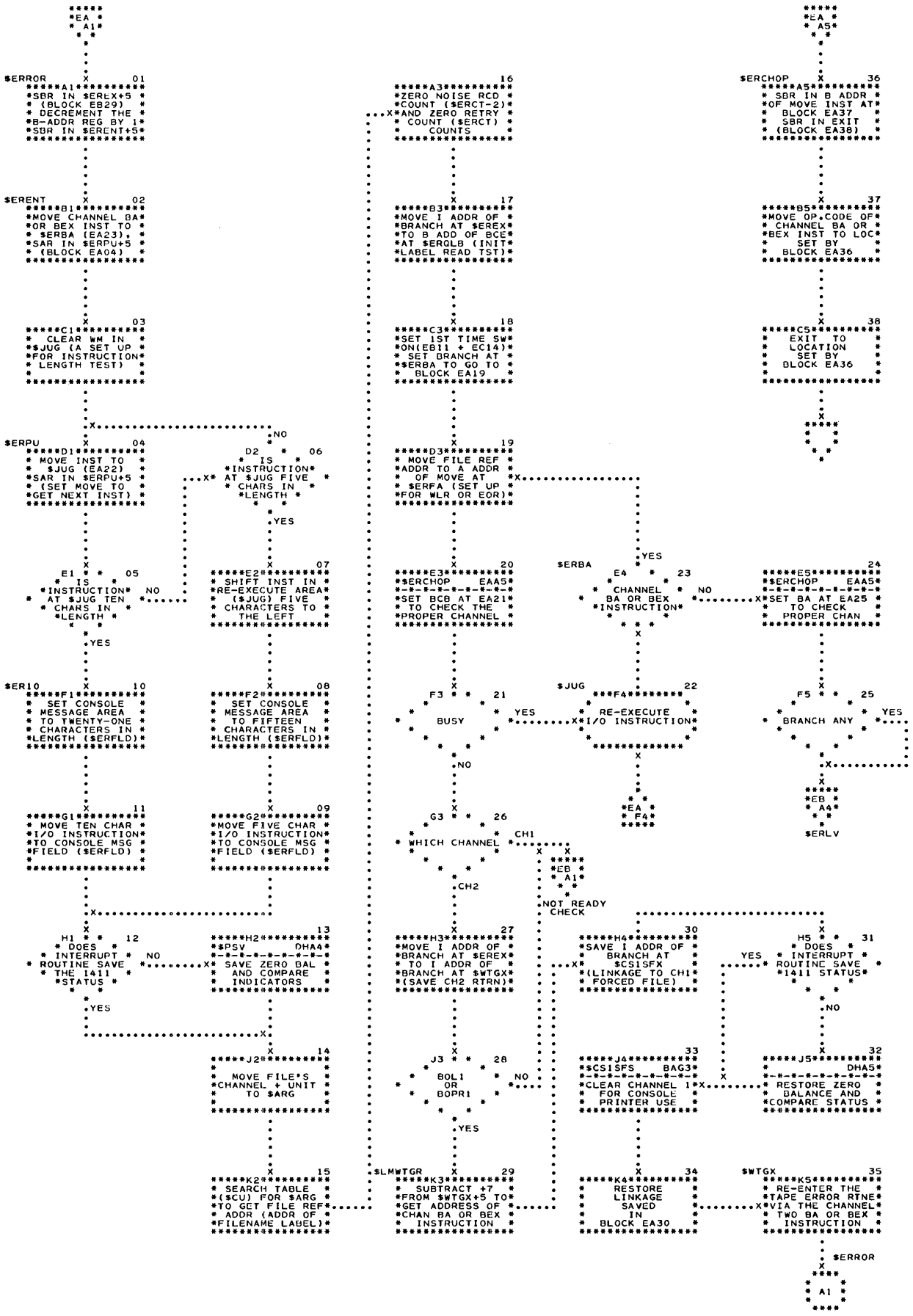


Chart EA. Tape Error Routine — Page 1 of 3

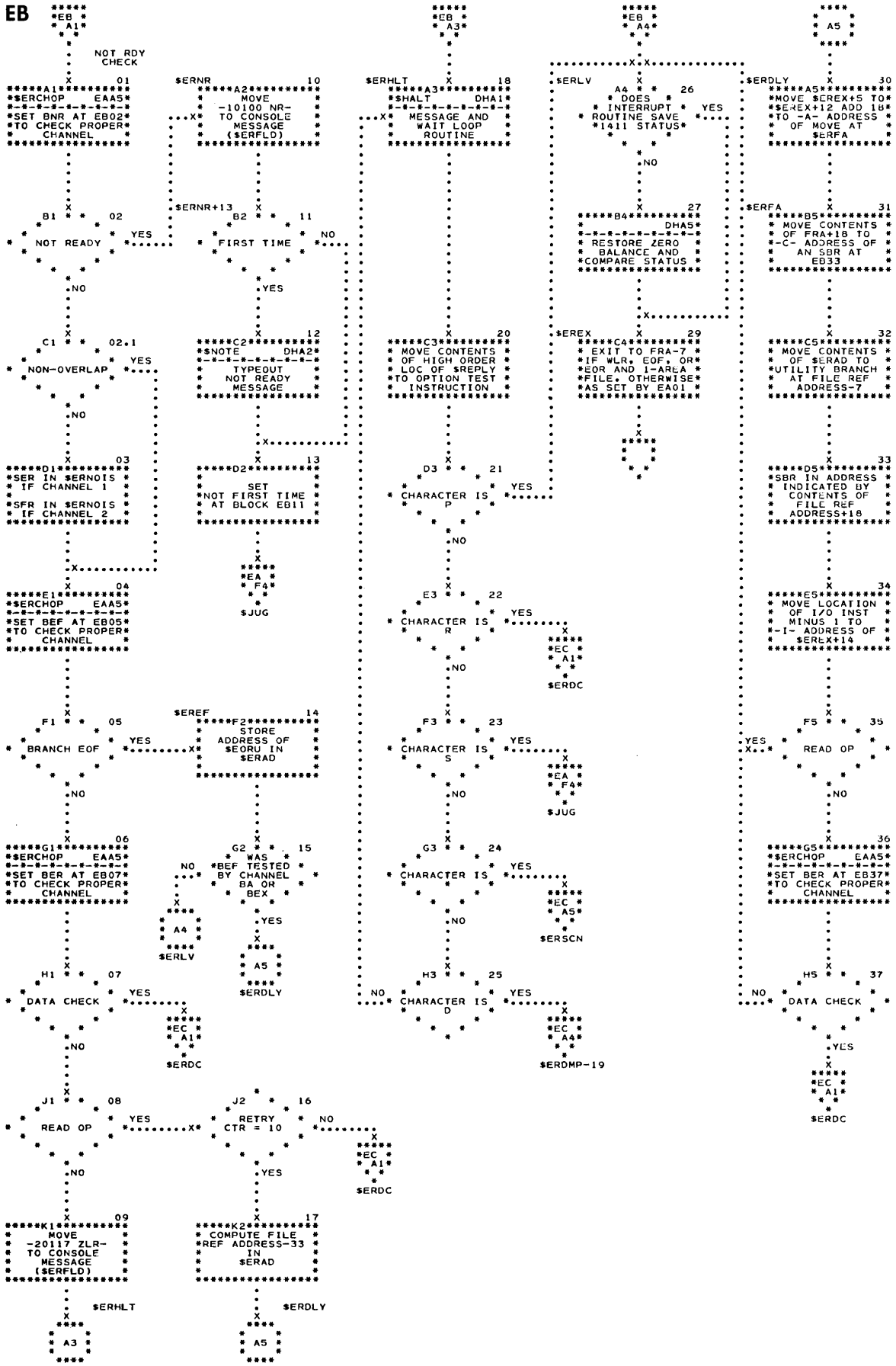


Chart EB. Tape Error Routine — Page 2 of 3

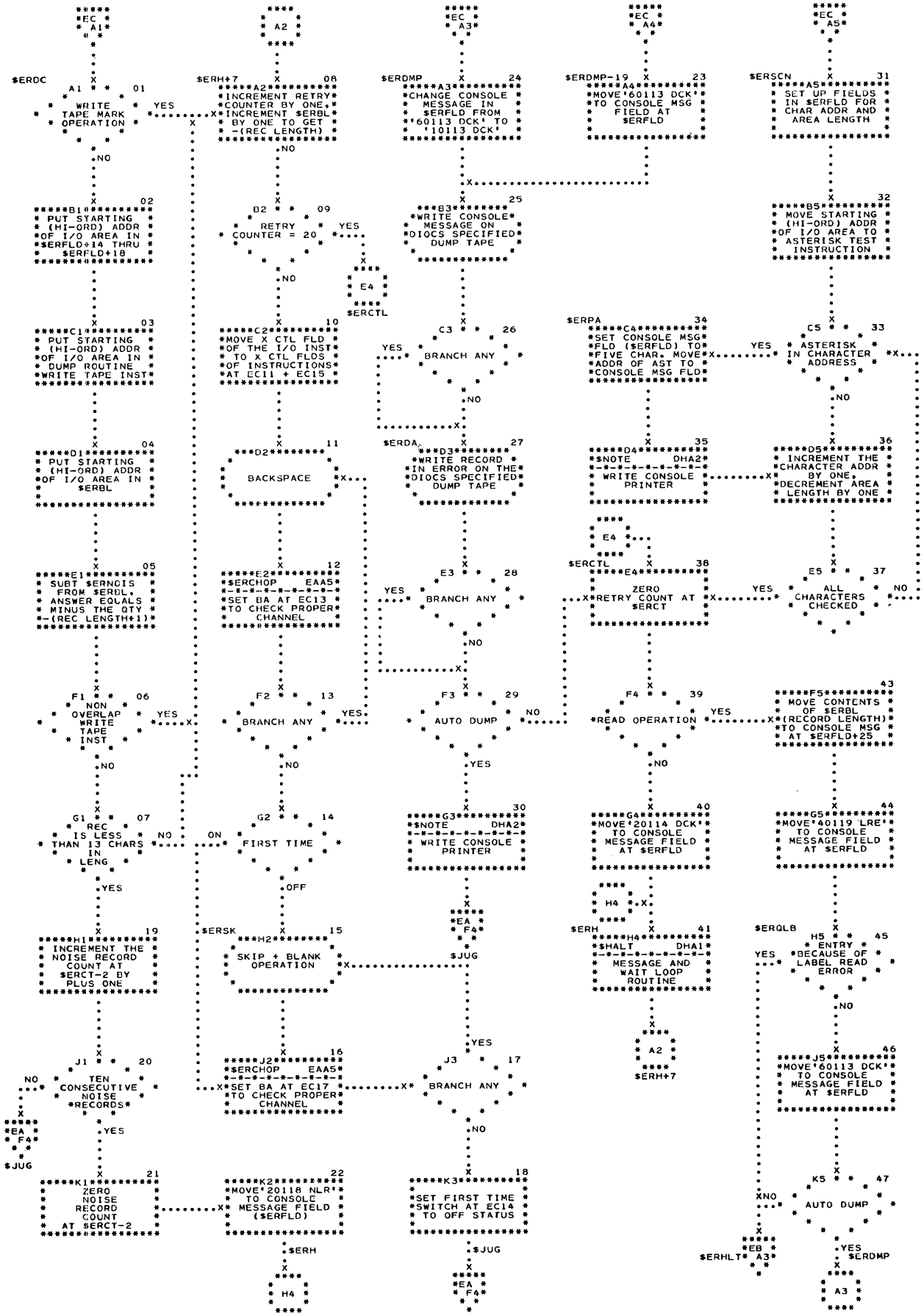


Chart EC. Tape Error Routine — Page 3 of 3

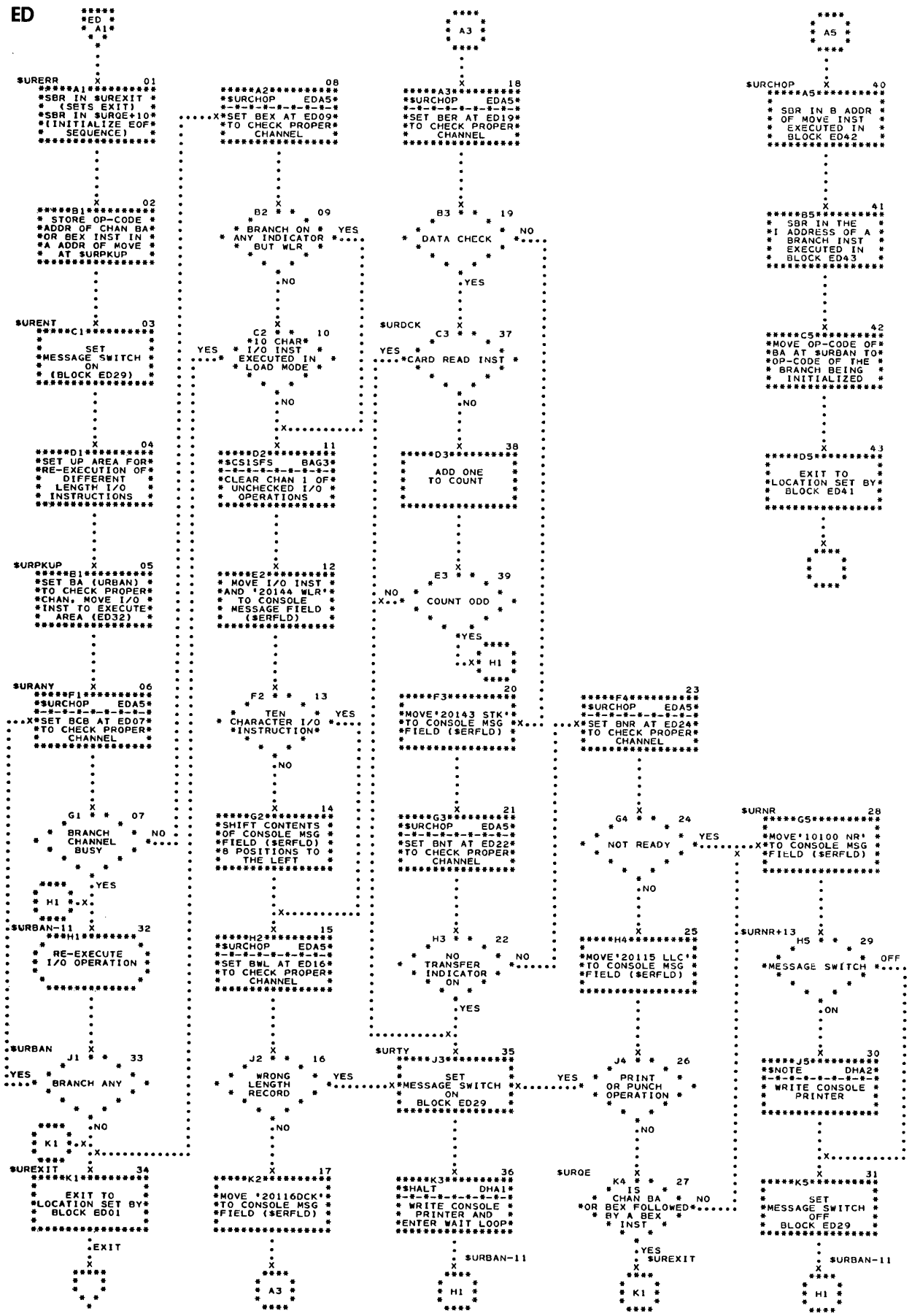


Chart ED. Unit Record Error Routine

Storage Map and Loading Sequence

The over-all storage requirements of IOCS may vary considerably from job-to-job, depending on the I/O specifications. These specifications determine the start of IOCS, the presence or absence of certain routines, the size of coding blocks within the routines, etc. Figure 6 shows the order of the IOCS routines by storage group in low-to-high storage sequence for a typical application.

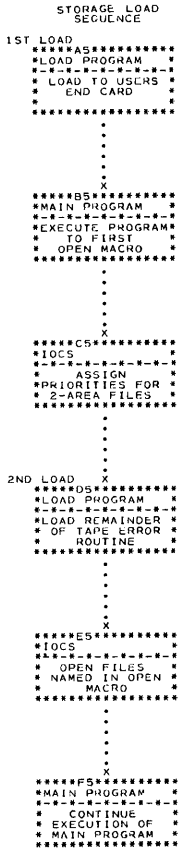
STORAGE MAP

SG	ROUTINE	LABEL	COMMENTS
1	PIVOT FOR RESTART	-----	LOCATIONS 00000 - 00024
2	INDEX WORDS	-----	LOCATIONS 00025 - 00099
3	INTERRUPT PIVOT	-----	LOCATIONS 00101 - 00115
4	STANDARD AUTOCODER	-----	LOCATIONS 00116 - 00349
5	PROCEDURE FOR AUTOMATIC TESTING	---	LOCATIONS 00350 - 00499
6	WHEN IOCS IS ORIGINED ABOVE LOCATION 500, THE LOCATIONS FROM 500 TO THE ORIGIN ARE AVAILABLE FOR THE MAIN PROGRAM	-----	LOCATIONS FROM 500 TO ORIGIN
7	CHANNEL SCHEDULERS	IOCSCS-ENT	EACH USES APPROXIMATELY 90 LOCATIONS
8	CONSOLE INTERRUPT SEQUENCE	IOCSCIPR	USES APPROXIMATELY 110 LOCATIONS
9	OPEN/CLOSE SUBROUTINE	IOCSCLOP	USES APPROXIMATELY 300 LOCATIONS
10	MESSAGE AND WAIT LOOP ROUTINE	IOCSHALT	USES APPROXIMATELY 125 LOCATIONS
11	CONSOLE PRINTER ERROR ROUTINE	IOCSCPERR	USES APPROXIMATELY 30 LOCATIONS
12	UNIT RECORD ERROR ROUTINE	IOCSURERR	USES APPROXIMATELY 430 LOCATIONS MAXIMUM
13	PRIORITY ASSIGNMENT ROUTINE	IOCSPAHSK	USES APPROXIMATELY 420 LOCATIONS, IT IS OVERLAYED BY A PORTION OF THE TAPE ERROR ROUTINE IF NON-OVERLAY HAS NOT BEEN SPECIFIED
14	RESIDENT PORTION OF THE TAPE ERROR ROUTINE	-----	OCCUPIES APPROXIMATELY 600 LOCATIONS
15	TAPE OPEN, CLOSE, AND END-OF-REEL ROUTINES	-----	OCCUPY APPROXIMATELY 500 LOCATIONS
16	LABEL ROUTINES	-----	OCCUPY APPROXIMATELY 900 LOCATIONS
17	1ST FILE SCHEDULER	IOCS-ENT	TAPE FILES USE UP TO 300 LOCATIONS UNIT RECORD FILES USE UP TO 150 LOCATIONS
18	1ST FILE TABLE		TAPE FILES USE UP TO 100 LOCATIONS
	SUCCEEDING FILE SCHEDULERS AND TABLES		
19	MAIN PROGRAM		

Figure 6. Storage Map

If the DIOCS PRIORITY entry of NONOVERLAY has not been specified, the IOCS routines are loaded in two distinct loads. The storage load sequence is shown in Figure 6.

When the non-overlay option is specified in the DIOCS, the IOCS priority assignment routine is needed in storage during running program time. In this instance, all of IOCS is loaded along with the main program.



Description of File Reference Table

Figure 7 illustrates the contents of a maximum file reference table. References 1 through 3 in the table are exceptional condition vectors; references 4 through 37 are known collectively as the file table. The structure of the table depends on the DIOCS entries (e.g., the DIOCS COUNTS entry governs the presence of references 24 and 25). When an entry is missing, the table is compressed. The number of file reference entries generated for each tape file is fixed according to the cumulative number of different entries for all files. The file type (or area indicator for tape), reference 5, is always generated; it is the only reference in the table which is fixed and not compressed; its location is always file reference address + 20; reference 5 is the only reference generated for unit record files.

The file reference tables correspond to an indexed label table. The indexed label table is a DA relative to zero and is referenced by index register 15. IOCS can process the file reference entries for any given file by loading index register 15 with the appropriate file reference address and then executing instructions that refer to the indexed labels. In Figure 7, each reference number is followed by the indexed label, if any. The actual prefixed label, if any, follows the indexed label.

The file reference table provides the information for interrogation by the various IOCS routines to control what procedures are performed. In general, a file reference may be modified by the user during program execution. Note, however, that the IOCS will refer to a file reference (counts excepted) only during the processing of OPEN, CLOSE, FEORL, and RDLIN macros and at end-of-reel. In some instances, to make a change effective, the file must be reopened. Some general comments follow to indicate which references may be changed.

References 1-3: The exceptional condition vectors may not be changed due to the structure of the file scheduler.

Reference 4: Priority may be changed only if the DIOCS PRIORITY is NONOVERLAY.

Reference 5: File type (or number of areas for tape) may not be changed due to the structure of the file scheduler.

Reference 6: May not be changed.

Reference 7: May be changed. In most cases, the file should be reopened.

Reference 8: May be changed.

Reference 9: IOCS linkage. May not be changed.

Reference 10: Input – May be changed. *Output* – May be changed to $CLSAB to prevent padding for a fixed, blocked file. The user must move a group mark/word mark to terminate a short block himself.

References 11-16: These label fields can be changed by programming and/or RDLIN cards.

Reference 17: May be changed to effect or skip label processing for a given reel.

Reference 18: May not be changed.

Reference 19: May be changed only in conjunction with references 7 and 8. The file may have to be reopened.

References 20-22: May be changed.

References 23-25: May be changed; however, it may cause a count discrepancy.

References 26-37: The exit indicators and/or exit addresses may be changed.

REF NO	INDEXEC LABEL	ACTUAL LABEL	OP CODE	OPERAND	FILE TYPE	KIND OF INDICATOR	NC FCS	DESCRIPTION OF INDICATOR AND CODES IF ANY
1			B	$EREX+7 $--TRIG	1-AREA 2-AREA	VECTOR VECTOR	7	LINKAGE PIVOT TO SCHEDULER FOR INPUT EOR
2	$PENSWE			$EREX+14 $--WTG	1-AREA 2-AREA	VECTOR VECTOR	7	LINKAGE PIVOT TO SCHEDULER FOR OUTPUT EOR
3	$TRIGEN		DCW	$EREX+5 $--TRIG+5	1-AREA 2-AREA	VECTOR VECTOR	5	LINKAGE CONSTANT USED BY ERROR ROUTINE FOR WLR AND EOR
4				N		PRIORITY	1	N= 0-9 IN HIGH TO LOW PRIORITY ORDER
5	$DTFACT	$--ACT		'N'	TAPE UR	NUMBER OF AREAS FILE TYPE	1	1= 1-AREA FILE, 2= 2-AREA FILE 3= READER FILE, 4= PUNCH FILE, 5= PRINTER FILE
6			DCW	' '			2	0= OPEN, BLANK= CLOSED (TAPE FILE ONLY)
7		$--BASE	DCW	'XXXNX'		BASE TAPE	5	MODE,PARITY,CHAN,AND UNIT FOR CURRENT BASE-E.G. M'U2X
8	$AD		DCW	'XXXNX'		ALT TAPE	5	MODE,PARITY,CHAN,AND UNIT FOR CURRENT ALT -E.G. M'U1X
9	$DTFI		DCW	$--INIT		FILE INIT ADDR	5	FOR INIT OF FILE SCHED DURING OPEN, FEORL, AND EOR
10	$DTFA	$--EOF	DCW	XX...XX $--PADS $CLSAB	INPUT OUTPUT OUTPUT	USER EOF ADDR PAD ROUTINE ADDR NO PADDING ADDR	5	ACCESSED WHEN REQUIRED BY THE EOR ROUTINE TO PAD PARTIAL BLOCK IF FIXED, BLOCKED FILE ON CLOSE DUMMY TO PRVENT PADDING WHEN NOT FIXED, BLOCKED
11	$D9	$--FSCK	DCW	'N'	INPUT OUTPUT	FILE SERIAL IND	1	0= CHECK, 1= NO CHECK 1= SET IT EQUAL TO TAPE SERIAL FOR 1ST REEL, 0= DONT
12	$HFS	$--HFS	DCW	'NNNNN-'	INPUT OUTPUT	FILE SERIAL NUMB	6	NNNNN= DTF SPECIFIED SERIAL NUMBER, OR BLANKS NNNNN= DTF SPECIFIED, OR TAPE SERIAL, OR BLANKS
13	$HRS		DC	'NNN '		REEL SEQ NUMBER	4	NNN= 001 OR DTF REEL SEQ, FOR 1ST REEL, UPDATED AT EOR
14	$HAN		DC	'X...X'		FILE NAME	10	HEADER NAME
15	$HCD		DC	'YYDDD'		CREATION DATE	5	YY= YEAR, DDD= DAY, DTF SPEC,OR FOR OUTPUT DATE IN 119
16	$HRC		DC	'-NNN '		RETENTION CYCLE	5	NNN = NUMBER OF DAYS, DTF SPECIFIED
17	$DTFLB	$--TFLB	DCW	'N'		LABEL TYPE	1	0= STANDARD, 1= NONE, 2= NON-STANDARD
18	$DTFL1	$--TFL1	DCW	'N'		FILE TYPE	1	0= OUTPUT, 1= INPUT
19	$DTFL2	$--TFL2	DCW	'N'		ALT DRIVE IND	1	0= NO ALTERNATE, 1= ALTERNATE IS SPECIFIED
20	$DTFL3	$--TFL3	DCW	'N'		LABEL CHECK IND	1	0= CHK COMPLETELY,1= NO CHK, 2= CHK FILE NAME ONLY
21	$DTFL4	$--TFL4	DCW	'N'		TM AFTER HDR IND	1	0= NO, 1=YES
22	$DTFL5	$--TFL5	DCW	'N'		REWIND IND	1	0= NO REWIND, 1= REWIND, 2= REWIND UNLOAD
23	$TBC	$--TBC	DCW	'NNNNN'		BLOCK COUNT	5	RUNNING BLOCK COUNT FOR CURRENT REEL
24	$TRC	$--TRC	DCW	'N...N'		RECORD COUNT	10	RUNNING RECORD COUNT FOR CURRENT REEL
25	$THT	$--THT	DCW	'N...N'		HASH TOTAL	16	RUNNING HASH TOTAL FOR CURRENT REEL
26	$D6 $D1	$--D6 $--D1	DCW DCW	'N' 'N'	INPUT OUTPUT	EXIT 6 IND EXIT 1 IND	1	0= NOT USED, 1=USED
27	$E6 $E1		DCW DCW	'X...X' 'X...X'	INPUT OUTPUT	EXIT 6 ADDRESS EXIT 1 ADDRESS	5	USED TO OVERRIDE OR IN LIEU OF STAND IDENTIFIER USED TO MODIFY STANDARD TRAILER
28	$D7 $D2	$--D7 $--D2	DCW DCW	'N' 'N'	INPUT OUTPUT	EXIT 7 IND EXIT 2 IND	1	0= NOT USED, 1= USED 0= NOT USED, 1= USED
29	$E7 $E2		DCW DCW	'X...X' 'X...X'	INPUT OUTPUT	EXIT 7 ADDRESS EXIT 2 ADDRESS	5	USED TO CHECK NON-STAND HEADER OR ADDITIONAL HEADER USED TO BUILD NON-STAND TRAILER OR ADD TRLR AFT STAND
30	$D3	$--D3	DCW	'N'	OUTPUT	EXIT 3 IND	1	0= NOT USED, 1= USED
31	$E3		DCW	'X...X'	OUTPUT	EXIT 3 ADDRESS	5	USED TO CHECK STANDARD HEADER AND/OR BUILD HEADER
32	$D4	$--D4	DCW	'N'	OUTPUT	EXIT 4 IND	1	0= NOT USED, 1= USED
33	$E4		DCW	'X...X'	OUTPUT	EXIT 4 ADDRESS	5	USED TO MODIFY STANDARD HEADER
34	$D5	$--D5	DCW	'N'	OUTPUT	EXIT 5 IND	1	0= NOT USED, 1= USED
35	$E5		DCW	'X...X'	OUTPUT	EXIT 5 ADDRESS	5	USED TO CHECK AND/OR BUILD NON-STAND AND/OR ADD HEADER
36	$D8	$--D8	DCW	'N'	OUTPUT	EXIT 8 IND	1	0= NOT USED, 1= USED
37	$E8		DCW	'X...X'	OUTPUT	EXIT 8 ADDRESS	5	

- Figure 7. File Reference Table

Appendix A — Glossary

ASSIGN: To modify and complete table entries and instructions that will be used by the running program.

BLOCK: One or more data records grouped to form one continuous record which will be written or read from tape, from or to storage.

BLOCKING FACTOR: The number of data records making up a tape record.

CHECKPOINT: A reference point at which error-free operation of the program has been verified and where the program may return for restart in the event of subsequent failure.

CHECKPOINT FILE: A tape record or records that contain the contents of storage and machine conditions necessary to restart a program at a checkpoint.

CHECKPOINT TAPE: The tape on which checkpoint records are written.

CLOSE: To terminate a file. For output files — write a tape mark and, if specified, an end-of-reel trailer and another tape mark. For both input and output files — rewind the tape and take a checkpoint, if specified.

CONTROL: The apparatus used to direct, guide, or restrain a mechanism or machine in operation. In computers, control is maintained through the sequence of instructions in a program. Control is often used in referring to the next instruction to be executed, by such phrases as, *control goes to,* or, *control branches to.*

DATA RECORD: A number of words of information grouped in a known manner which will be used as data for a given operation.

EOF(END OF FILE): The logical end of an organized collection of information directed toward some purpose. For multiple-reel files, it is recognized at the end of the last reel.

EOR (END OF REEL): The end of all records on a single tape. The trailer on labeled input tapes contains information defining end of reel. EOR on unlabeled input tapes must be recognized by a user's routine. The user is able to identify an end of reel by recognizing the last record of a reel. EOR for output files is normally indicated by recognition of the end-of-tape reflector.

FIXED LENGTH DATA RECORDS: Data records within a tape file, all of which contain the same number of characters.

FORCE CONDITION: An indication that there is no record in the read-in area available for processing or that the write-out area is unavailable to receive information. When this condition exists, IOCS will take steps to fill the read-in area with information or to execute a write from the write-out area, thus freeing it for processing.

HASH COUNT (HASH TOTAL): The cumulative total of the sets of characters in a hash field for all records, (in a file, in a reel, or that are processed in a particular way). The total is made for auditing or control purposes.

HASH FIELD: The position in a record from which hash counts or hash totals are derived.

INITIALIZATION: The resetting of counters, switches, and addresses at specified times in a program.

IOCS (INPUT/OUTPUT CONTROL SYSTEM): A program developed to handle all necessary unit record or tape input and output procedures to relieve programmers of duplicating their efforts for most programs they write.

LABEL: A record or records, written on tape, containing identifying information concerning the file on the tape. For specifications, see IOCS bulletin.

LINKAGE: A series of instructions which enable a transfer to and return from one program routine to another.

MACRO: An open-ended sequence of machine instructions produced by a processor on recognition of a source-language statement. These instructions can perform a function defined by the parameters given in the source statement. They may consist, in part, of a linkage to a closed subroutine.

MULTI-FILE REEL: A tape reel which contains more than one tape file.

NOISE RECORD: A redundant non-data pulse which is picked up by the read head.

PARAMETER: A quantity left unspecified at some stage of an operation and to which the user may assign arbitrary values. Also, a field in the operand of a macro statement. It may be given different names or values which allow the macro generator to generate machine instructions that have the correct address, index words, etc., for a large variety of programming situations.

PRIME: To fill input areas.

RECORD COUNT: A count of the number of records in a file, in a reel, or that are processed in a particular way.

RETENTION CYCLE: The number of calendar days following the creation date that a file is to be saved if standard headers are used.

REQUEST: The I/O operations which the main program seeks to perform in IOCS.

SUBROUTINE: A small routine that can be included as part of several larger routines. Two major types exist;

1. Open — This routine is inserted directly, wherever needed, in such a way that control enters and exits in a sequential manner.
2. Closed — A routine which occurs only once, non-sequentially, in a program. It may have several entry and exit points. It is entered and left via linkage.

TAPE RECORD: The information contained between two successive inter-record gaps.

VARIABLE LENGTH DATA RECORDS: Data records within a tape file, at least two of which do not contain the same number of characters.

VARIABLE LENGTH TAPE RECORDS: Data tape records in a file that contain variable length data records, or data tape records in a tape file at least two of which have different blocking factors.

Appendix B — List of Abbreviations

ACPT	ACCEPT
ADDR	ADDRESS
AFT	AFTER
AR	AREA
AST	ASTERISK
AUTO	AUTOMATIC
BAL	BALANCE
BKSP	BACKSPACE
BLKCNT	BLOCK COUNT
BLKD	BLOCKED
CHAN	CHANNEL
CHAR	CHARACTER
CHK	CHECK
CHKPT	CHECKPOINT
CLS	CLOSE
CNT	COUNT
COND	CONDITION
CORR	CORRECTED
CTR	COUNTER
CTL	CONTROL
CU	CHANNEL AND UNIT
CW	CLEAR WORD MARK INSTRUCTION
D-MOD	d-MODIFIER
EOR	END-OF-REEL
EXT	EXTERIOR
F.S.	FILE SCHEDULER
FLD	FIELD
FORC	FORCING
FORCD	FORCED
FRA	FILE REFERENCE ADDRESS
G PR	GREATER PRIORITY
FXD	FIXED
GM	GROUP MARK
H-O	HIGHER ORDER
HDR	HEADER
HGHST	HIGHEST
HI	HIGH
HI-ORD	HIGH ORDER
I/O	INPUT/OUTPUT
IDENT	IDENTIFICATION
IND	INDICATOR
INFO	INFORMATION
INIT	INITIALIZATION
INPT	INPUT
INST	INSTRUCTION
INTRPT	INTERRUPT
L-O	LOW ORDER
LBL	LABEL
LENG	LENGTH
LOC	LOCATION
MSG	MESSAGE
MVE	MOVE
NEC	NECESSARY
NO.	NUMBER
NOP	NO OPERATION
NORM	NORMAL
OP	OPERATION
OP-CODE	OPERATION CODE
OVRLP	OVERLAP
P	PARAMETER
PA	PRIORITY ALERT
PNCH	PUNCH
PNDG	PENDING
POS	POSITION
PREV	PREVIOUS
PRI	PRIORITY
PRIM	PRIME
PROCD	PROCEDURE
PROG	PROGRAM
PT	POINT
PTR	PRINTER
QTY	QUANTITY
RCD	RECORD
RCP	READ CONSOLE PRINTER INSTRUCTION
RD	READER, READ
REF	REFERENCE
REG	REGISTER
REJT	REJECT
REQ	REQUIRED
RET	RETURN
RTN	ROUTINE
RWD	REWIND
RWU	REWIND AND UNLOAD
SCHD	SCHEDULER
SCHED	SCHEDULER
SKP	SKIP
SPEC	SPECIFIED
STG	STORAGE
SUB	SUBSTRACT
SW	SWITCH
TM	TAPE MARK
UR	UNIT RECORD
USUL	USUAL
VAR	VARIABLE
WLR	WRONG LENGTH RECORD
WM	WORD MARK
WR	WRITE
X CTL	X-CONTROL
XREG	INDEX REGISTER

Appendix C — Cross Reference Indexes

Three cross references are included to provide quick access to specific points on the flow chart.

Part 1: The connector cross reference lists the off-page entry connector under ENTRY CONN, the off-page exit connectors associated with that entry connector under EXIT CONN, and the associated block number under BLOCK NUMBER. For example, DBA2 under ENTRY CONN is an off-page entry connector on Chart DB; DCK1 and DCK3 under EXIT CONN are the chart locations which have off-page exit connectors to DBA2; and DC22 and DC13 under BLOCK NUMBER are the respective block numbers.

Part 2: The subroutine cross reference lists the subroutine name under SUBROUTINE; the entry connector to the subroutine, if any, under ENTRY CONN; and each block which represents the subroutine under BLOCK LOCATION and BLOCK NUMBER. For example, $WTMRU under SUBROUTINE and DGB5 under ENTRY CONN indicates that DGB5 is the off-page entry connector to the detail chart of the $WTMRU subroutine; DBJ2 and DEB4 under BLOCK LOCATION, and DB33 and DE27 under BLOCK NUMBER are the chart locations and block numbers, respectively where the subroutine is represented.

Part 3: The label cross reference lists the symbolic or actual label under LABEL, the chart location(s) where the label occur(s) under BLOCK LOCATION, and the block number(s) under BLOCK NUMBER. For example, $INTEXT under LABEL is shown at chart locations AAK1 and BAH1 under BLOCK LOCATION and at block numbers AA23 and BA20 under BLOCK NUMBER.

ENTRY CONN	EXIT CONN	BLOCK NUMBER
DBA1	DAF3	DA24
	DEK4	DE39
DBA2	BEK4	BE10
	BEK5	BE20
	DCK1	DC22
	DCK3	DC13
DBC2	DCH4	DC40
DBC4	DCC1	DC03
	DCC2	DC06
	DCK3	DC13
DBK4	DDK3	DD22
	DEK2	DE33
DCA1	DBB3	DB14
DDA1	BBF1	BB11
	BBF5	BB31
	BCE3	BC20
	BDD3	BD16
DED1	DDD1	DD25
	DDK1	DD33
DEG2	DDK4	DD21

ENTRY CONN	EXIT CONN	BLOCK NUMBER
DFB2	DAD4	DA26
DFB3	CBG5	CB36
	DBK4	DB25
	DCG4	DC37
DGD4	DFF2	DF07
EAF4	EBD2	EB13
	EBF3	EB23
	ECG3	EC30
	ECJ1	EC20
	ECK3	EC18
EBA1	EAG3	EA26
EBA3	ECK5	EC47
EBA4	EAF5	EA25
ECA1	EBE3	EB22
	EBH1	EB07
	EBH5	EB37
	EBJ2	EB16
ECA4	EBH3	EB25
ECA5	EBG3	EB24

SUBROUTINE NAME	ENTRY CONN	BLOCK LOC	BLOCK NUMBER
		ABC2	AB03
		ABK4	AB15
		BBG2	BB14
		BCH3	BC28
		DBF2	DB30
		DBF4	DB19
		DCE4	DC43
		DCJ5	DC37
		DDH4	DD19
		DDJ1	DD32
		DEF4	DE25
		DEH5	DE19

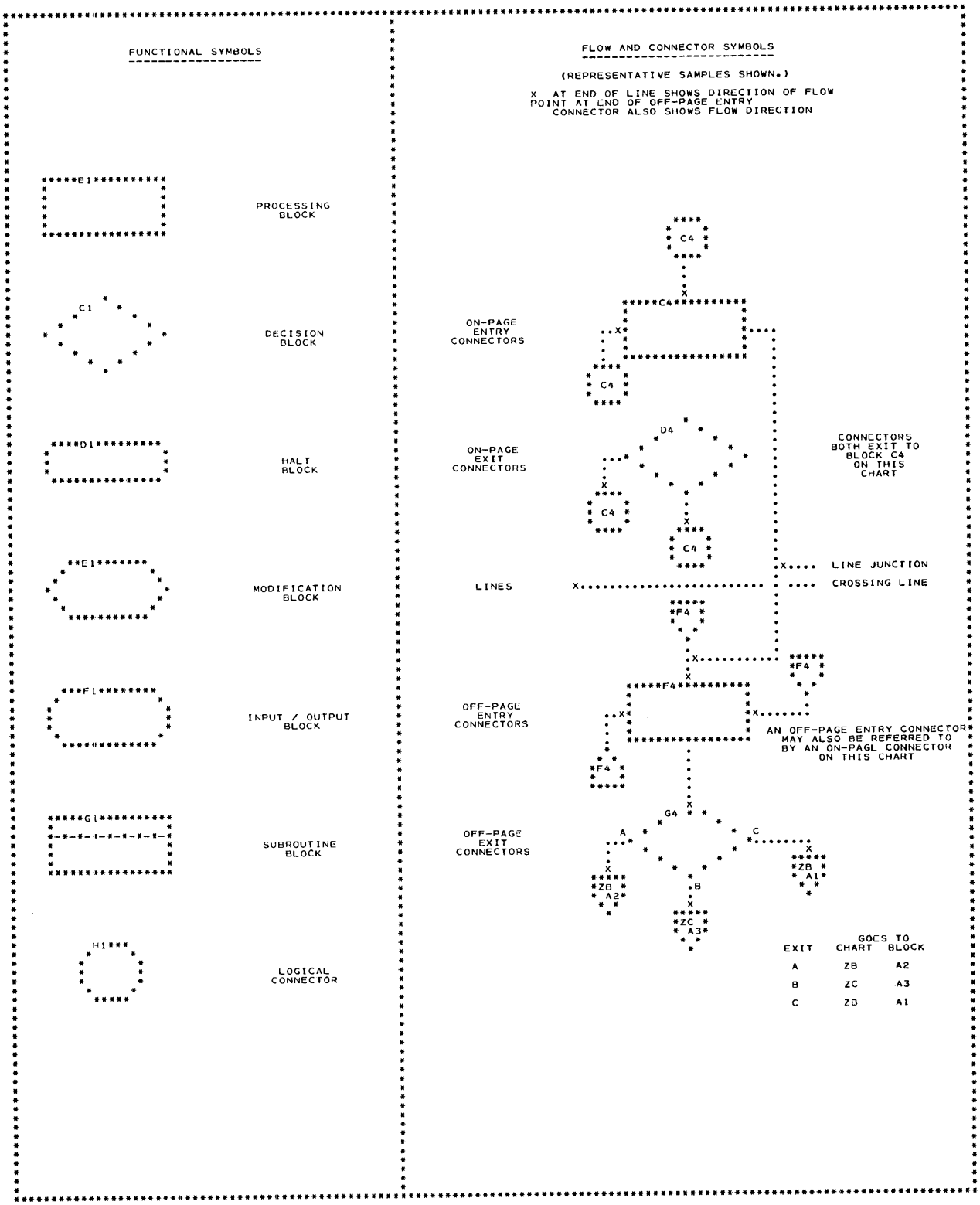
SUBROUTINE NAME	ENTRY CONN	BLOCK LOC	BLOCK NUMBER
	DHA5	EAJ5	EA32
		EBB4	EB27
	EAA1/EDA1	CCE4	CC34
$--INIT	BEA4/A5	DBK1	DB10
$--PADS	BFB2/B4	CAE5	CA25
		DEB1	DE03
$--WTG	BDD1	DDG2	DD29
		DEB5	DE12
$CHKPT		DBJ5	DB24
$CLOP	DFB1	ABB3	AB01
$CLSA+7	DEA3	ABC4	AB21
$CPERR	DHG2	CCD2	CC24
		DHD2	DH14
$CS-RET	F2/K3	AAJ4	AA45
$CS-RET	BAB1/F1	CCE2	CC05
$CS-SCN	BAE4/D1	CCJ3	CC12
$CS-SFS	BAG3/D1	CBC1	CB03
		CBD4	CB24
		CCB1	CC01
		CCB4	CC31
$CS1SFS	BAG3	ACA1	AC09
		CCB3	CC21
		CCB5	CC41
		DFC4	DF22
		EAJ4	EA33
		EDD2	ED11
$CS2SFS	BAD1	CCC5	CC42
	BAD1	DFD4	DF23
$ENTA	DBA1	ABE4	AB08
$ENTAB	DBA2	ABF4	AB09
$ENTC	DCA1	ABG4	AB10
$ENTD	DBC4	ABG3	AB11

SUBROUTINE NAME	ENTRY CONN	BLOCK LOC	BLOCK NUMBER
$ENTF	DBC2	ABG5	AB12
$ENTH	DLG2	ADD4	AB22
$ENTJ	DBK4	ABH4	AB13
$ENTRY	AAA2	AAD5	AA36
	PAA3	BCE2	BC21
		CBG1	CB10
		CBH4	CB28
		CCE3	CC25
		CCF5	CC51
		CCG2	CC14
		CCH4	CC37
$EORU	DDA1	ABB4	AB19
$ERCHOP	EAA5	EAE3	EA20
		EAE5	EA24
		EDA1	EB01
		EBE1	EB04
		EBG1	ED06
		EBG5	EB36
		ECE2	EC12
		ECJ2	EC16
$ERROR	EAA1	AAJ5	AA46
		BBF1	BB11
		BBF5	BB31
		BCA4	BC30
		BDD4	BD26
		CCF2	CC07
		DGE5	DG33
		DGG2	DG29
		DGH2	DG08
$EXIT	DFB2	ABB2	AB02
$EXITRU	DFB3	ABJ4	AB14
$HALT	DHA1	ACE3	AC19
		ACH3	AC25
		DCH1	DC20
		DCJ3	DC12
		DEK3	DE36
		DGG5	DG36
		EGA3	EB18
		ECH4	EC41
		EDK3	ED36
$IPEOR	DDA5	ABC5	AB20
$NOTE	DHA2	ACK5	AC29
		DCE3	DC08
		DCG3	DC10
		DDF5	DD11
		DDH5	DD13
		DDK5	DD15
		DEK4	DE39
		DHB1	DH02
		EBC2	EB12
		ECD4	EC35
		ECG3	EC30
		EDJ5	ED30
$PAEXIT	DAC4	ABF2	AB05
$PAHSK	DAB3	ABE3	AB07
	DAB4	ABE2	AB04
$PSV	DHA4	EAH2	EA13
$RDLIN	DGD4	ABB1	AB24
$READRU	DGB1	DBH5	DB22
		DCB1	DC02
		DDA5	DD06
$REENT	DFG5	DBG2	DB31
		DBG4	DB20
		DCF4	DC44
		DCK5	DC38
		DDJ4	DD20
		DDK1	DD33
		DEG4	DE26
		DEJ5	DE20
		DFF4	DF25

SUBROUTINE NAME	ENTRY CONN	BLOCK LOC	BLOCK NUMBER
$RWDRU	DGB3	DBA4	DB13
		DCG4	DC39
		DEG1	DE29
$RWURU	DGB4	DEH1	DE31
$STLE	DFB4	DDB1	DD02
		DFC1	DF02
$SWBX	DFB5	DBE2	DB29
		DBE4	DB18
		DCD4	DC42
		DCH5	DC36
		DDG4	DD18
		DDH1	DD31
		DEE4	DE24
		DEG5	DE18
		DFE3	DF14
$TPCLOS	DEA2	ABC3	AB18
$URCHOP	EDA5	EDA2	ED08
		EDA3	ED18
		EDF1	ED06
		EDF4	ED23
		EDG3	ED21
		EDH2	ED15
$URERR	EDA1	CBE2	CB06
		CBG4	CB27
		CBG5	CB36
$WRITRU	DGB2	DCH4	DC40
		DEK5	DE21
$WTMRU	DGB5	DBJ2	DB33
		DEB4	DE13
		DEG3	DE27
CHAN SCHED	BAA3	BBH1	BB07
	BAB5	BBJ4	BB29
		BDD2	BD17
	BAE3	BCE5	BC34
		BDF5	BD28
	BAG4	BBC1	BB03
		BBD4	BB24
		BCD1	BC04
		BDF1	BD06
F.S.	BBC4/BCB1	BFE1	BF07
		BFH3	BF20
F.S.	BCA1/BBB1	CAB4	CA13
F.S.	BCA5/BDD5	BAD5	BA10
F.S.	BDA1/BBB4	CAG2	CA08
FILE SCHED		ABK3	AB17
	AAF4	AAE2	AA11
		AAJ2	AA20
	AAH5	AAC2	AA06
		AAG2	AA15
FORM 4 RCD ONLY		BBH2	BB15
FORM 4 WLR SEQ		BBH3	BB16
		BCH5	BC35
IOCS		RRC5	
		RRE5	
LOAD PROGRAM		ABG2	AB06
		RRA5	
		RRD5	
MACRO		ABK5	AB16
MAIN PROGRAM		RRB5	
		RRF5	
PNDG SW NETWORK		BAH4	BA22
SCHED	B3/F3	AAC4	AA39
SCHED	C3/G3	AAK4	AA46

BLOCK LABEL	BLOCK LOC	BLOCK NUMBER
$--ACT+11	CBB5	CB31
$--BYP	BCD2	BC19
$--EMTY	AAA4	AA31
	BBB1	BB01
	BCA1	BC01
	CBB1	CB02
$--EXIT	AAF5	AA38
	BBK1	BB09
	BBK4	BB30
	BCH2	BC24
	BDF2	BD19
	CBJ1	CB12
	CBJ4	CB29
$--FULL	BBB4	BB21
	BDA1	BD01
	CBB4	CB22
$--INIT	BEA4	BE01
	BEA5	BE11
$--IOA	BBD1	BB04
	BBE4	BB25
	BCJ1	BC09
	BDB4	BD22
	CBD1	CB04
	CBE4	CB25
$--IOB	BCK1	BC10
	BDB5	BD23
$--PA	AAA5	AA33
	BCA2	BC12
	BDA2	BD10
$--PADS	BFB2	BF01
	BFB4	BF11
$--PFOR	BFG3	BF19
$--PLB1	BFG2	BF06
	BFJ4	BF18
$--PLB2	BFD2	BF03
	BFF4	BF15
$--SA	BCA3	BC15
	BDA3	BD13
$--SFX	CBC1	CB03
	CDD4	CB24
$--TRIG	AAC5	AA35
	BBH1	BB07
	BDH4	BB28
	BCE3	BC20
	BDD3	BD16
	CBG1	CB10
	CDH4	CB28
$--WLR	BBH3	BB16
	BCH5	BC35
$--WTG	BCB1	BC02
	BDD1	BD04
$--0000N	BCG1	BC07
	BCJ1	BD09
$ATTN	BAB4	BA02
$BCRT	DDE4	DD16
$CLOP	DFB1	DF01
$CLSA	DEF2	DE08
$CLSA+7	DEA3	DE09
$CLSABA	DEC2	DE04
$CLSB	DED5	DE15
$COMP	DAC1	DA05
$CPERR	DHG2	DH21
$CPEX	DHJ2	DH24
$CS1ENT	AAB1	AA04
	BAC4	BA05
$CS1PR	AAC1	AA05
	BAD4	BA06
$CS1RET	AAF2	AA12
	BAB1	BA17
$CS1SCN	BAE4	BA07
$CS1SF	BAE3	BA13
$CS1SFS	AAB3	AA07
	BAG3	BA11
$CS1SFX	BAG2	BA16
$CS1S3	BAG4	BA09
$CS2ENT	AAF1	AA13
	BAC1	BA18
$CS2PR	AAG1	AA14
$CS2RET	AAK3	AA21
	BAF1	BA19
$CS2SFS	AAF3	AA16
$DATER	DCJ2	DC18
$END	DAC2	DA13
$ENTA	DBA1	DB01
$ENTAB	DBA2	DB11
$ENTC	DCA1	DC01
$ENTCR	DCB1	DC02
$ENTD	DBC4	DB16
$ENTE	DCG4	DC39
$ENTF	DBC2	DB27
$ENTG	DDK3	DD22
$ENTH	DEG2	DE28
$ENTI	DBJ4	DB23
$ENTJ	DBK4	DB25
$ENTRY	AAA2	AA01
	BAA3	BA03
$EORU	DDA1	DD01
$ERBA	ACA5	AC04
	EAE4	EA23
$ERCHOP	EAA5	EA36
$ERCTL	ACF4	AC22
	ECE4	EC38
$ERDA	ECD3	EC27
$ERDC	ACD2	AC17
	ECA1	EC01
$ERDLY	ACC4	AC13
	EBA5	EB30
$ERDMP	ECA3	EC24
$ERDMP-19	ECA4	EC23
$EREF	EBF2	EB14
$ERENT	EAB1	EA02
$EREX	EDC4	EB29
$ERFA	EDB5	EB31
$ERH	ACE3	AC19
	ECH4	EC41
$ERH+7	ACE4	AC20
	ECA2	EC08
$ERHLT	ACH3	AC25
	EBA3	EB18
$ERLV	ACC5	AC06
	EBA4	EB26
$ERNR	ACB4	AC11
	EBA2	EB10
$ERNR+13	EBB2	EB11
$ERPA	ECC4	EC34
$ERPU	EAD1	EA04
$ERQLB	ACG4	AC23
	ECH5	EC45
$ERROR	ACA2	AC01
	EAA1	EA01
$ERSCN	ACK3	AC30
	ECA5	EC31
$ERSK	ECH2	EC15
$ER10	EAF1	EA10
$EXIT	DFB2	DF03
$EXITRU	DFB3	DF11
$FINIT	DBJ1	DB09
$HALT	DHA1	DH01
$HALTX	DHH1	DH08
$IN	DAE3	DA23
$INTEXT	AAK1	AA23
	BAH1	BA20
$IPEOR	DDA5	DD06
$JUG	ACA4	AC03
	EAF4	EA22
$LBIN	DGE1	DG04
$LBOP	DGG1	DG06
$LINK	DAH1	DA10
$LMWTGR	ACB1	AC08
	EAK3	EA29
$MOVE	DAJ1	DA11
$NHL	DCE1	DC23
$NIH	DCJ3	DC12
$NOH	DCH1	DC20
$NOTE	DHA2	DH11
$OPEOR	DDD1	DD25
$OPEORA	DED1	DE05
$OPHD	DCE2	DC14
$OPHDAB	DCA4	DC28
$OUT	DAF2	DA16
$PAEXIT	DAC4	DA25
$PAHSK	DAB3	DA02
	DAB4	DA01
$PRIMER	DEB4	DE13
$PRIME3	DEA5	DE11
$PSV	DHA4	DH31
$PSX	DHJ4	DH39
$RDLIN	DGD4	DG31
$READRU	DGB1	DG01
$REENT	DFG5	DF41
$RWDB	DGF3	DG27
$RWDRU	DGB3	DG21
$RWDRUA	DGC3	DG24
$RWURU	DGB4	DG22
$STLE,$SCS	DFB4	DF21
$STLEXT	DFJ5	DF43
$SUSXA	DEC4	DE22
$SWBX	DFB5	DF31
$SWBXA	DFD5	DF33
$TPCLOS	DEA2	DE01
$URANY	EDF1	ED06
$URBAN	EDJ1	ED33
$URBAN-11	EDH1	ED32
$URCHOP	EDA5	ED40
$URDCK	EDC3	ED37
$URENT	EDC1	ED03
$URERR	EDA1	ED01
$UREXIT	EDK1	ED34
$URNR	EDG5	ED28
$URNR+13	EDH5	ED29
$UROPEN	DFG1	DF51
$URPKUP	EDE1	ED05
$URQE	EDK4	ED27
$URTY	EDJ3	ED35
$WRITRU	DGB2	DG11
$WTGX	EAK5	EA35
$WTMRU	DGB5	DG23
00101	BAA4	BA01
1ST LOAD	RRA5	
2ND LOAD	RRD5	

Appendix D — Sample Autochart Symbols



Appendix E — DIOCS-Generated Label Definitions

This section lists the labels in IOCS that are generated because of DIOCS entries. With each label is an explanation of the routine or data area the label addresses.

```
IOCSAD      - ALTERNATE DRIVE.
                 RELATIVE POSITION OF THE FILE SCHEDULERS ALTERNATE
              DRIVE INFORMATION.

IOCSARG     - ARGUMENT.
                 WHEN THE I/O COMMAND IN ERROR IS PROCURED, THE ERROR
              ROUTINE USES THE CHANNEL SYMBOL AND UNIT NUMBER TO LOOK
              UP AGAINST THE CHANNEL UNIT TABLE TO OBTAIN THE
              FILE SCHEDULER ADDRESS.

IOCSATTN    - ATTENTION.
                 SEQUENCE OF INSTRUCTIONS FOR STORING THE PROGRAM
              STATUS LATCHES PRIOR TO ENTRY INTO IOCS.

IOCSBCRT    - BRANCH CHECK - REWIND TAPE.
                 MOVE EXIT 6 ADDRESS TO THE SWITCH BOX, TURN OFF THE
              PENDING SWITCH FOR THE FILE, BRANCH TO EXIT 6 ROUTINE,IF
              SPECIFIED, AND CONTINUE TO THE RWD/RWU SEQUENCE, IF AT
              END OF REEL.

IOCSBLANK   - BLANKS.
                 BLANKS USED TO CLEAR THE LABEL AREA BEFORE CONSTRUC-
              TION OF HEADER LABELS.

IOCSCCTAB   - CHANNEL CHANGE TABLE AB.
                 TABLE CONTAINING CONSTANTS NECESSARY TO CHANGE A FILE
              SCHEDULER TO THE ALTERNATE CHANNEL ACCORDING TO THE CON-
              TENTS OF THE FILE SCHEDULER CHANNEL INDICATOR DURING THE
              OPEN.

IOCSCHKPT   - CHECKPOINT.
                 CHECKPOINT SEQUENCE - ENTERED VIA THE CHKPT MACRO IN-
              STRUCTION OR DURING THE END-OF-REEL SEQUENCE AT IOCSENTI.
              WILL WRITE THE CHECKPOINT IDENTIFICATION HEADER AND THE
              CHECKPOINT RECORD.

IOCSCIA     - CONSOLE INQUIRY AREA.
                 AREA USED TO STORE INDEX REGISTERS AND PROGRAM STATUS
              LATCHES DURING AN INQUIRY REQUEST INTERRUPT.

IOCSCIE     - CONSOLE INQUIRY ENTRY.
                 BRANCH TO THE USER'S INQUIRY ROUTINE. THE USER'S ROUTINE
              MUST HAVE A SBR INSTRUCTION AS ITS FIRST COMMAND TO ROUTE
              THE RETURN FOR RESTORATION OF THE PROGRAM STATUS LATCHES
              AND INDEX REGISTERS.

IOCSCIPR    - CONSOLE INQUIRY PRIORITY ROUTINE.
                 ROUTINE STORING PROGRAM LATCHES AND INDEX REGISTERS
              THAT MAY BE USED IN THE USER'S INQUIRY ROUTINE.

IOCSCKBSP   - CHECKPOINT BACK SPACE.
                 BACKSPACE INSTRUCTIONS FOR ERRORS WHEN WRITING CHECK-
              POINT RECORDS ONLY. IF DURING THE PROCESS OF WRITING A
              CHECKPOINT, A REFLECTIVE SPOT IS ENCOUNTERED, IOCS
              WILL BACKSPACE OVER BOTH CHECKPOINT AND HEADER RECORD AND
              FORCE END OF REEL. IOCS WILL THEN OPEN THE NEXT FILE
              AND PROCEED TO WRITE THE HEADER AND CHECKPOINT RECORD.

IOCSCKEXIT  - CHECKPOINT EXIT.
                 EXIT FROM THE CHECKPOINT ROUTINE TO THE USER'S PROGRAM.

IOCSCKHLDA  - CHECKPOINT HOLD AREA.
                 HOLD AREA, DURING CHECKPOINT WRITING, FOR MEMORY LOCA-
              TIONS 00001 THRU 00024. THESE LOCATIONS WILL BE STORED
              HERE PRIOR TO WRITING OF THE CHECKPOINT RECORD.

IOCSCKMSG   - CHECKPOINT MESSAGE.
                 CONSOLE INDICATION AFTER THE CHECKPOINT RESTART HAS
              BEEN SUCCESSFULLY COMPLETED. '20186 RST'.

IOCSCKREC   - CHECKPOINT RECORD.
                 INFORMATION THAT WILL BE WRITTEN AS THE HEADER OF EACH
              CHECKPOINT WRITTEN. THIS INFORMATION IS USED BY THE RE-
              START PROGRAM TO FIND THE LOCATIONS OF INFORMATION NECES-
              SARY TO RESTORE THE PROGRAM TO A PRIOR STATUS.

IOCSCKRCC   - CHECKPOINT COMPLETE - COUNT.
                 CONSOLE INDICATION THAT A CHECKPOINT HAS BEEN WRITTEN.
              LAST THREE POSITIONS OF THE MESSAGE CONTAIN THE SEQUEN-
              TIAL NUMBER IDENTIFYING THIS CHECKPOINT. '10185 CPT XXX'.

IOCSCLOP    - CLOSE OPEN.
                 BEGINNING OF THE ROUTINE FOR OPEN, CLOSE, FEORL, AND
              RDLIN MACRO INSTRUCTIONS.

IOCSCLOSD   - CLOSE DUMP.
                 DUMP TAPE CLOSE SEQUENCE - ENTERED VIA THE CLOSD MA-
              CRO.

IOCSCLOSDD  - CLOSE DUMP DUMMY.
                 THE CONTROL UNIT INSTRUCTION THAT IS MODIFIED BY THE
              OPERAND OF THE CLOSD MACRO.

IOCSCLOSDX  - CLOSE DUMP EXIT.
                 DUMP TAPE CLOSE EXIT TO RETURN TO THE USER'S PROGRAM.

IOCSCLSA    - CLOSE A.
                 MODIFY AN OUTPUT FILE SCHEDULER, IF A 2-AREA FILE, FOR
              RETURN TO IOCS AT END OF REEL.

IOCSCLSAB   - CLOSE AB.
                 TESTS FOR A FEORL, IF NOT, WILL PROCEED TO CLEAR THE
              REEL SEQUENCE NUMBER.

IOCSCLSB    - CLOSE B.
                 ASSEMBLE AND WRITE TRAILER LABEL - EXIT 1 SEQUENCE
              CONSTRUCTS THE TRAILER LABELS IN THE LABEL AREA. SETS
              UP FOR THE POSSIBILITY OF AN EXIT 1 ROUTINE AND WRITES
              THE TRAILER LABEL.

IOCSCOMP    - COMPARE.
                 SEQUENCE USED TO DETERMINE IF THE FILE CURRENTLY BEING
              OPENED IS ALREADY IN THE SORT TABLE.
```

```
IOCSCPERR  - CONSOLE PRINTER ERROR ROUTINE.
             TESTS ALL LATCHES EXCEPT WRONG-LENGTH RECORD AND NOT
             READY.

IOCSCPEX   - CONSOLE PRINTER ERROR EXIT.
             EXIT FROM THE CONSOLE ERROR ROUTINE.

IOCSCSNENT - CHANNEL SCHEDULER N ENTRY.
             ENTRY USED DURING 1405/1301 IOCS OPERATIONS. NOT NEC-
             ESSARY FOR CARD/TAPE OPERATIONS ALTHOUGH SEQUENCE IS GEN-
             ERATED.

IOCSCSXA1  - CHANNEL SCHEDULER X AREA 1.
             AREA USED TO STORE INDEX REGISTERS AND PROGRAM STATUS
             LATCHES DURING A UNIT RECORD PRIORITY INTERRUPT.

IOCSCSXENT - CHANNEL SCHEDULER X ENTRY.
             ENTRY POINT INTO THE CHANNEL SCHEDULER ROUTINE. WILL
             TEST FOR OVERLAP IN PROCESS ON THIS CHANNEL AND BYPASS
             ANY TESTING OF THE LATCHES ON THIS CHANNEL IF IN OVERLAP.

IOCSCSXE1  - CHANNEL SCHEDULER ENTRY 1.
             BRANCH TO USER'S UNIT RECORD PRIORITY ROUTINE. USER'S
             ROUTINE MUST INCORPORATE A SBR INSTRUCTUON AS THE FIRST
             COMMAND TO ROUTE THE RETURN TO IOCS FOR RESTORATION
             OF PROGRAM STATUS LATCHES AND INDEX REGISTERS.

IOCSCSXPR  - CHANNEL SCHEDULER X PRIORITY REQUEST.
             A TEST OF THE CHANNEL PRIORITY REQUEST LATCH FOR THE
             CAUSE OF INTERRUPT. THE I-FIELD OF THIS INSTRUCTION WILL
             CONTAIN THE ADDRESS OF THE INTERRUPTING FILE SCHEDULERS
             IOCSXXBA - STATUS INDICATOR TEST.

IOCSCSXRET - CHANNEL SCHEDULER X RETURN.
             POINT OF RETURN FOR ALL FILE SCHEDULERS ON THE CHANNEL
             AFTER AN I/O OPERATION HAS BEEN STARTED. THE STORE B REG-
             ISTER COMMAND WILL TRIGGER IOCSCXPR TO BRANCH TO A STATUS
             TEST ON THE NEXT INTERRUPT.

IOCSCSXSCN - CHANNEL SCHEDULER X SCHEDULER CLEAR NOP.
             FILE SCHEDULER ENTRY POINT AFTER ERROR CHECK FOR I/O
             COMMAND JUST COMPLETED.  PENDING I/O OPERATIONS MAY BE
             STARTED. THIS IS ALSO USED TO START THE FORCING OF THE
             CHANNEL WHEN IT IS NECESSARY TO COMPLETE ONE I/O COMMAND
             BEFORE STARTING A SECOND. THE FIRST GET AFTER AN OPEN
             MACRO WILL CAUSE THIS CONDITION.

IOCSCSXSF  - CHANNEL SCHEDULER X SCHEDULER FORCE.
             STARTING OF THE CHANNEL CLEARING BY FORCING THE PROC-
             ESS TO THE ERROR ROUTINE IF OVERLAP IS IN PROCESS ON
             THIS CHANNEL.

IOCSCSXSFS - CHANNEL SCHEDULER X SCHEDULER FORCE STATUS.
             WHEN A FILE SCHEDULER IS CONFRONTED WITH THE CONDITION
             THAT ANOTHER I/O OPERATION HAS TO BE STARTED FOR THE SAME
             FILE BEFORE THE PREVIOUS OPERATION IS STARTED OR COM-
             PLETED, THE FILE SCHEDULER WILL BRANCH HERE TO FORCE THE
             CHANNEL TO CLEAR.

IOCSCSXSFX - CHANNEL SCHEDULER X SCHEDULER FORCE EXIT.
             BRANCH BACK TO THE FILE SCHEDULER THAT FORCED THE
             CLEARING OF THIS CHANNEL.

IOCSCSXS3  - CHANNEL SCHEDULER X SCHEDULER 3.
             A BRANCH EXIT PRIORITY ALERT TO THE HIGH PRIORITY FILE
             SCHEDULERS PENDING SWITCH. WILL START THE TEST OF ALL
             FILE SCHEDULERS ON THE CHANNEL FOR PENDING OPERATIONS.

IOCSCSXUPR - CHANNEL SCHEDULER X UNIT PRIORITY ROUTINE.
             ROUTINE FOR STORING PROGRAM LATCHES AND INDEX REGIS-
             TERS THAT MAY BE USED DURING THE USER'S UNIT RECORD IN-
             TERRUPT ROUTINE.

IOCSCU     - CHANNEL UNIT.
             A 140-POSITION TABLE CONSISTING OF SEVEN-POSITION DCW'S
             CONTAINING A DTF ADDRESS, CHANNEL, AND UNIT OF EVERY TAPE
             FILE OPENED BY THE PROGRAM. USED BY THE ERROR ROUTINE TO
             DETERMINE WHICH FILE SCHEDULER INITIATED THE ERROR.

IOCSDATER  - DATE ERROR.
             RETENTION PERIOD ERROR SEQUENCE. SET UP THE DATE ERROR
             MESSAGE.

IOCSDTFA   - DTF ADDRESS.
             THE RELATIVE LOCATION OF THE FILE SCHEDULER END-OF-
             FILE ADDRESS.

IOCSDTFACT - DTF ACTIVITY.
             RELATIVE LOCATION OF THE FILE SCHEDULER ACTIVITY
             DIGIT.

IOCSDTFBX  - DTF BOX
             WILL CONTAIN THE DTF ADDRESSES USED AS PARAMETERS FOR
             THE OPEN, CLOSE, FEORL, AND RDLIN MACROS. A ONE-POSITION
             CHARACTER BEFORE AND AFTER THE DTF BOX WILL CONTAIN THE
             MACRO IDENTIFIER EXPLAINED UNDER IOCSEXIT.

IOCSDTFI   - DTF INITIALIZER.
             RELATIVE LOCATION OF AN AREA WITHIN THE FILE SCHEDULER
             CONTAINING ITS INITIALIZATION ADDRESS.

IOCSDTFLB  - DTF LABEL.
             RELATIVE LOCATION OF THE LABEL TYPE CHECK CHARACTER.

IOCSDTFL1  - DTF LABEL 1.
             RELATIVE LOCATION OF THE FILE I/O TYPE CHARACTER.

IOCSDTFL2  - DTF LABEL 2.
             RELATIVE LOCATION OF THE ALTERNATE DRIVE CHECK CHARAC-
             TER.

IOCSDTFL3  - DTF LABEL 3.
             RELATIVE LOCATION OF THE LABEL CHECK CHARACTER.

IOCSDTFL4  - DTF LABEL 4.
             RELATIVE LOCATION OF THE HEADER T/M CHECK CHARACTER.

IOCSDTFL5  - DTF LABEL 5.
             RELATIVE LOCATION OF THE REWIND OPTION CHECK CHARACTER
```

IOCSD1 - DIGIT 1.
RELATIVE LOCATION OF THE EXIT 1 CHECK CHARACTER.

IOCSD2 - DIGIT 2.
RELATIVE LOCATION OF THE EXIT 2 CHECK CHARACTER.

IOCSD3 - DIGIT 3.
RELATIVE LOCATION OF THE EXIT 3 CHECK CHARACTER.

IOCSD4 - DIGIT 4.
RELATIVE LOCATION OF THE EXIT 4 CHECK CHARACTER.

IOCSD5 - DIGIT 5.
RELATIVE LOCATION OF THE EXIT 5 CHECK CHARACTER.

IOCSD6 - DIGIT 6.
RELATIVE LOCATION OF THE EXIT 6 CHECK CHARACTER.

IOCSD7 - DIGIT 7.
RELATIVE LOCATION OF THE EXIT 7 CHECK CHARACTER.

IOCSD8 - DIGIT 8.
RELATIVE LOCATION OF THE EXIT 8 CHECK CHARACTER.

IOCSD9 - DIGIT 9.
RELATIVE LOCATION OF THE FILE SERIAL CHECK CHARACTER WITHIN THE FILE SCHEDULER.

IOCSEND - END.
LOOKS UP AGAINST THE SORT TABLE FOR THE PROPER POSITION TO INSERT THE FILE BEING OPENED.

IOCSENTA - ENTRY A.
COMMON FILE SCHEDULER INITIALIZATION SEQUENCE. MOVES THE DTF ADDRESS TO THE TAPE ASSIGNMENT TABLE. USES THE DTF ADDRESS TO FIND THE FILE INITIALIZATION ADDRESS, THEN PROCEEDS TO EXECUTE THE INITIALIZATION.

IOCSENTAB - ENTRY AB.
COMMON RE-ENTRY POINT FROM ALL TAPE FILE INITIALIZATION ROUTINES. TESTS FOR REWIND PRIOR TO OPEN AND STARTS THE STANDARD LABEL TESTS.

IOCSENTC - ENTRY C.
INPUT/OUTPUT COMMON TESTS.
EXIT THREE SEQUENCE - OUTPUT.
TESTS FOR A USER EXIT 3 ROUTINE. MOVES THE EXIT 3 ADDRESS TO THE SWITCH BOX AND READS THE OUTPUT HEADER. THEN IT WILL EXECUTE THE USER EXIT 3 ROUTINE. IF THERE IS NO EXIT 3 ADDRESS AND THERE IS NO LABEL CHECKING, IOCS WILL BYPASS LABEL READING ON AN OUTPUT FILE.

IOCSENTCR - ENTRY CR.
CALLS FOR READING OF ALL INPUT LABELS AND ALL OUTPUT LABELS NOT CONTROLLED BY AN EXIT 3 ROUTINE OR NO CHECK. IF INPUT LABEL IS NOT TO BE CHECKED, CONTROL IS RETURNED TO IOCSENTD. IF THE LABEL IS TO BE CHECKED OR WRITTEN, A COMPARE IS MADE FOR A 'IHDR ' AND AN ERROR OCCURS IF NOT TRUE. IF AN INPUT LABEL IS TO BE CHECKED, PROCESSING CONTINUES TO CHECK CREATION DATE, FILE SEQUENCE, IDENT., AND SERIAL NUMBER, IF NECESSARY, ETC. IF THE INPUT HEADER IS INVALID, AN ERROR MESSAGE IS CONSTRUCTED AND TYPED.

IOCSENTD - ENTRY D.
EXIT 7 SEQUENCE - INPUT. MOVES EXIT 7 ADDRESS TO THE SWITCH BOX AND TESTS IF THERE IS AN EXIT 7 ROUTINE. EXAMINES THE DTF TABLE FOR A TAPE MARK AFTER THE HEADER.

IOCSENTE - ENTRY E.
ROUTINE FOR REWINDING THE FILE PRIOR TO AND WRITING OF THE CONSTRUCTED LABEL.

IOCSENTF - ENTRY F.
EXIT 5 SEQUENCE. MOVES EXIT 5 ADDRESS TO THE SWITCH BOX AND CHECKS FOR A USER'S EXIT 5 ROUTINE. CHECKS THE DTF TABLE FOR THE POSSIBILITY OF A WRITE TAPE MARK AFTER THE OUTPUT HEADER.

IOCSENTG - ENTRY G.
END-OF-FILE EXIT SEQUENCE. MOVE THE FILE SCHEDULERS END-OF-FILE ADDRESS TO IOCSEXIT.

IOCSENTH - ENTRY H.
TEST THE REWIND OPTION AND UPDATE REEL SEQUENCE NUMBER BY ONE. IF A FEORL, BRANCH TO THE REEL CHANGE SEQUENCE AND SET UP CONSOLE MESSAGE CONCERNING A REEL CHANGE.

IOCSENTI - ENTRY I.
TESTS FOR A FEORL AND BRANCHES TO WRITE A CHECKPOINT RECORD.

IOCSENTJ - ENTRY J.
SETS THE FILE SCHEDULER BEING OPENED INTO A NOT-PENDING STATUS IF THE FILE HAS TWO I/O AREAS.

IOCSENTRY - ENTRY.
SEQUENCE USED, AFTER ALL MACRO EXECUTIONS, TO TEST IOCS FOR PENDING OPERATIONS PRIOR TO RETURNING TO THE USER'S PROGRAM.

IOCSEORU - END-OF-REEL ROUTINE.
INITIALIZATION FOR END OF REEL. SETS FOR FEORL, CLEARS THE CHANNELS AND TESTS FOR AN OUTPUT FILE. FOR AN INPUT FILE, EXIT 6 ADDRESS IS TESTED.

IOCSERAD - ERROR ADDRESS.
ADDRESS OF THE DTF IN ERROR DURING WRONG-LENGTH RECORD CHECK. WILL BE DECREMENTED BY 33 FOR POSITIONING OF THE DTF ROUTINE TO OBTAIN THE WRONG-LENGTH RECORD ADDRESS, IF ANY.

IOCSERBA - ERROR BRANCH ANY.
WORK AREA FOR THE STATUS TEST OF THE I/O COMMAND INITIATING THE ERROR SEQUENCE.

IOCSERBL - ERROR BAD LOCATION.
MULTI-PURPOSE AREA.
1. WILL CONTAIN THE CONTENTS STORED BY THE SER OR SFR FOR NOISE RECORD CHECK.
2. DURING A DATA CHECK ERBL WILL CONTAIN THE RELATIVE POSITION OF THE CHARACTER IN ERROR WITHIN THE I/O AREA.

```
IOCSERCHOP - ERROR CHANNEL - OPERATION.
             SEQUENCE FOR MODIFICATION OF OPERATION CODES ON STATUS
           TESTS USED BY THE ERROR ROUTINE ACCORDING TO THE CHANNEL.

IOCSERCT   - ERROR COUNT.
             CONSISTS OF TWO 2-POSITION DCW'S USED AS ERROR COUNTS.
           THE FIRST, OR UNLABELED, DCW IS FOR CONTROLLING THE NUM-
           BER OF RETRIES ON A NOISE RECORD READ - 10 ATTEMPTS. THE
           SECOND, OR LABELED DCW CONTROLS THE NUMBER OF RETRIES
           INITIATED BY A DATA CHECK BEFORE AN ERASE FORWARD IS GIV-
           EN - 20 ATTEMPTS.

IOCSERCTL  - ERROR CONTROL.
             TESTING THE TYPES OF DATA CHECKS.
               1. LABEL READ ERROR. '40119 LRC'.
               2. READ DATA CHECK.  '60113 DCK'.
               3. WRITE DATA CHECK. '20114 DCK'.

IOCSERDA   - ERROR DUMP AREA.
             WRITING ERROR I/O AREA ON THE DUMP TAPE.

IOCSERDC   - ERROR DATA CHECK.
             SEQUENCE FOR TESTING IF THE ERROR IS A NOISE RECORD OR
           A DATA CHECK. '20118 NLR'.

IOCSERDLY  - ERROR DELAY.
             MOVES NORMAL ERROR EXIT TO MAKE THE EXIT AVAILABLE
           FOR WRONG-LENGTH RECORD ERROR ROUTINES SPECIFIED BY THE
           USER.

IOCSERDMP  - ERROR DUMP.
             DUMP TAPE - WRITING OF IOCSERFLD TO IDENTIFY THE FOL-
           LOWING ERROR RECORD.

IOCSEREF   - ERROR END FILE.
             SET UP OF THE ERROR EXIT TO BRANCH TO THE END-OF-REEL
           ROUTINE AT AN END OF FILE.

IOCSEREX   - ERROR EXIT.
             EXIT FROM THE ERROR ROUTINE TO THE FILE SCHEDULER FOR
           THE WRONG-LENGTH RECORD ROUTINE OR THE COMPLETION OF THE
           I/O ROUTINE.

IOCSERFA   - ERROR FUTURE ADDRESS.
             MODIFIES AND INITIALIZES THE FILE SCHEDULER WHEN THE
           ERROR IS A WRONG-LENGTH RECORD ERROR.

IOCSERFH   - ERROR FUTURE HOLD.
             MOVES THE DTF ADDRESS OF THE FILE IN ERROR FOR THE
           MODIFICATION AND USE OF THE FILE SCHEDULER WHEN THE ERROR
           IS A WRONG-LENGTH RECORD. '20117 ZLR'.

IOCSERFLD  - ERROR FIELD.
             A 29-POSITION DCW FOLLOWED BY A GROUP MARK
           WORD MARK FOR HOLDING CONSTRUCTED ERROR MESSAGES FOR
           IOCS INITIALLY '20183 CI                          ',G.

IOCSERH    - ERROR HALT 1.
             SEQUENCE TO TYPE ERROR MESSAGE, UPDATE AND TEST ERROR
           COUNTS, AND SET UP OF AN ERASE FORWARD.

IOCSERHLT  - ERROR HALT 2.
             ROUTINE FOR TESTING OF THE CONSOLE REPLY RECEIVED FROM
           THE CONSOLE OPERATOR FOR THE OPTION DESIRED.

IOCSERLV   - ERROR LEAVE.
             EXIT FROM THE ERROR ROUTINE TO THE FILE SCHEDULER FOR
           THE WRONG-LENGTH RECORD ROUTINE OR THE COMPLETION OF THE
           I/O ROUTINE.

IOCSERNOIS - ERROR NOISE.
             THE WORK AREA USED TO STORE THE E OR F REGISTER ACCORD-
           ING TO THE CHANNEL DETECTING THE ERROR. USED TO CHECK
           FOR NOISE RECORDS ONLY IF A VALID WRONG-LENGTH RECORD.

IOCSERNR   - ERROR NOT READY.
             NOT READY MESSAGE ROUTINE AND LOOP. '10100 NR '.

IOCSEROPTN - ERROR OPTION.
             A DCW CONTAINING THE FIRST CHARACTER OF THE VARIOUS
           REPLIES THAT MAY BE REQUESTED BY THE ERROR ROUTINE.

IOCSERPA   - ERROR PRINT ADDRESS.
             MODIFYING OF THE IOCSERFLD WHEN AN ASTERISK IS DETECTED
           DURING THE SCAN. MOVES THE MEMORY ADDRESS OF THE AS-
           TERISK FOR TYPING WITH THE ERROR MESSAGE.

IOCSERPU   - ERROR PICK UP.
             SEQUENCE FOR FINDING THE I/O COMMAND CAUSING THE ERROR.

IOCSERQLB  - ERROR QUESTION LAST BRANCH ANY.
             TEST TO CHECK IF THE ERROR WAS GENERATED DURING THE
           LABEL READ/WRITE ROUTINE.

IOCSERROR  - ERROR.
             TAPE ERROR ROUTINE SEQUENCE.

IOCSERSCN  - ERROR SCAN.
             SEQUENCE FOR SCANNING THE ERROR I/O AREA FOR DETECTION
           OF THE ASTERISKS REPLACING THE INVALID CHARACTERS.

IOCSERSK   - ERROR SKIP.
             ERASE FORWARD.

IOCSERIO   - ERROR IO.
             MOVES THE I/O COMMAND IF THE COMMAND IS TEN POSI-
           TIONS IN LENGTH TO IOCSJUG.

IOCSEXIT   - EXIT.
             MOVES THE DTF ADDRESS OF THE TAPE FILE BEING OPERATED
           UPON TO THE DTF BOX ALONG WITH THE OP. CODE IDENTIFIER
           THAT DEFINES THE MACRO TYPE.
               C DEFINES OPEN.
               + DEFINES RDLIN.
               * DEFINES FEORL.
               ) DEFINES CLOSE.
               J DEFINES THE END OF MACRO.

IOCSEXITRU - EXIT ROUTINE.
             ROUTINE TESTS FOR THE COMPLETION OF THE OPEN, CLOSE,
           FEORL, AND RDLIN MACROS. IF NOT, CAUSES A BRANCH TO
           BRING IN THE NEXT PARAMETER - DTF ADDRESS.
```

IOCSE1 - EXIT 1.
RELATIVE LOCATION OF THE EXIT 1 ADDRESS.

IOCSE2 - EXIT 2.
RELATIVE LOCATION OF THE EXIT 2 ADDRESS.

IOCSE3 - EXIT 3.
RELATIVE LOCATION OF THE EXIT 3 ADDRESS.

IOCSE4 - EXIT 4.
RELATIVE LOCATION OF THE EXIT 4 ADDRESS.

IOCSE5 - EXIT 5.
RELATIVE LOCATION OF THE EXIT 5 ADDRESS.

IOCSE6 - EXIT 6.
RELATIVE LOCATION OF THE EXIT 6 ADDRESS.

IOCSE7 - EXIT 7.
RELATIVE LOCATION OF THE EXIT 7 ADDRESS.

IOCSE8 - EXIT 8.
RELATIVE LOCATION OF THE EXIT 8 ADDRESS.

IOCSGM - GROUP MARK.
BLANK FOLLOWED BY A GROUP MARK WORD MARK USED BY IOCS IN THE CONSTRUCTION OF ITS VARIOUS MESSAGES AND AREAS

IOCSHALT - HALT.
CONSOLE REPLY WAITING LOOP. WAITING FOR THE OPERATOR TO REPLY TO AN IOCS ERROR MESSAGE.

IOCSHALTX - HALT EXIT.
BRANCH BACK TO IOCS ROUTINE WHICH DETERMINED THE ERROR. USUALLY THE ROUTINE RETURNED TO WILL TEST THE OPERATOR REPLY FOR THE ACTION TO BE TAKEN.

IOCSHAN - HEADER ALPHABETIC NAME.
RELATIVE LOCATION OF THE HEADER IDENTIFICATION.

IOCSHCD - HEADER CREATION DATE.
RELATIVE LOCATION OF THE HEADER CREATION DATE.

IOCSHDBL - HEADER BLANKS.
BLANKS USED TO CLEAR THE LABEL AREA BEFORE CONSTRUCTION OF HEADER LABELS.

IOCSHFS - HEADER FILE SERIAL.
RELATIVE LOCATION OF THE HEADER FILE SERIAL NUMBER WITHIN THE FILE SCHEDULER.

IOCSHITBL - HIGH TABLE.
HIGH-ORDER POSITION OF THE PENDING-SWITCH SORT TABLE. 140 POSITIONS CONSISTING OF 7-CHARACTER SEGMENTS CONTAINING THE UNITS POSITION OF THE FILE PENDING SWITCH. CHANNEL SYMBOL. AND PRIORITY NUMBER.

IOCSHRS - HEADER REEL SEQUENCE.
RELATIVE LOCATION OF THE HEADER REEL SEQUENCE.

IOCSIN - IN.
UPDATES THE HIGHER PRIORITY PENDING SWITCH OPERAND TO THE ADDRESS OF THE CURRENT OPENING FILE'S PENDING SWITCH. RESETS INDEX REGISTER 15 TO THE DTF ADDRESS CONTAINED IN THE DTF BOX.

IOCSINTEXT - INTERRUPT EXIT.
THE BRANCH ENTER PRIORITY ALERT TO THE USER'S PROGRAM AFTER AN IOCS RELEASE.

IOCSIPEOR - INPUT END OF REEL.
STANDARD INPUT TRAILER LABEL SEQUENCE. READS THE TRAILER LABEL AND CHECKS HASH TOTALS. RECORD AND BLOCK COUNT. AN UNEQUAL COMPARISON CAUSES AN ERROR MESSAGE.

IOCSJUG - ERROR JUG.
WORK AREA FOR I/O COMMAND INITIATING THE ERROR SEQUENCE.

IOCSLBA - LABEL AREA.
EIGHTY-POSITION AREA FOR BUILDING OR READING STANDARD LABELS.

IOCSLBAREA - LABEL AREA.
EIGHTY-POSITION AREA FOR BUILDING OR READING STANDARD LABELS.

IOCSLBIN - LABEL INPUT.
MOVING INFORMATION FROM THE DTF TO THE LABEL I/O COMMAND. CHANNEL. UNIT. AND CHANNEL STATUS OPERATION CODE.

IOCSLBOP - LABEL OUTPUT.
READ/WRITE COMMAND TO OR FROM THE LABEL AREA. STATUS TEST FOR A NOT READY. CHANNEL BUSY. OR DATA CHECK. BRANCHES BACK TO THE CONTROLLING ROUTINE.

IOCSLINK - LINKAGE.
MODIFIES OPERANDS OF PENDING SWITCHES IF CURRENT FILE HAS BEEN OPENED PREVIOUSLY. CAUSES AN OPENING OR MISSING LINK IN THE PENDING SWITCH CHAIN.

IOCSLMWTGR - LAST MINUTE WAITING ROUTINE.
ENTERED WHEN AN ERROR ENCOUNTERED ON CHANNEL 2 WILL CAUSE A MESSAGE TO BE TYPED ON THE CONSOLE PRINTER WHILE CHANNEL 1 IS STILL IN OVERLAP PROCESS.

IOCSMDC - MOVE D CONTROL.
INITIALIZATION FOR AN ALTERNATE DRIVE. SET UP FILE SCHEDULER WITH THE ALTERNATE DRIVE BECOMING THE PRIMARY DRIVE AND VICE VERSA.

IOCSMOVE - MOVE.
MODIFIES THE PENDING SWITCH SORT TABLE TO CORRESPOND TO THE CHANGE MADE IN THE PENDING SWITCH CHAIN. OPENING THE SORT TABLE FOR A NEW INSERTION.

IOCSNEPO - OPEN - (SPELLED BACKWARDS).
UNITS POSITION OF THE TABLE CONTAINING THE ADDRESS OF VARIOUS OPEN AND CLOSE ROUTINES RELATING TO THE FILE ACTIVITY CODE.

```
IOCSNHL     - NO HEADER LABEL.
                 MOVES UNIT INFORMATION ON AN ERROR CAUSED BY AN INVALID
              HEADER WHEN THE FILE SPECIFIES STANDARD HEADERS.
              SETS UP BOTH THE NO INPUT AND NO OUTPUT HEADER ERROR
              MESSAGES. '40130 NOH', AND '30133 NIH'.

IOCSNIH     - NO INPUT HEADER.
                 SEQUENCE FOR WRITING THE INPUT INFORMATION IF IN ERROR
              ON THE CONSOLE TYPEWRITER. THE ROUTINE ALSO TESTS THE
              OPERATOR REPLY.

IOCSNOH     - NO OUTPUT HEADER.
                 TYPES THE OUTPUT HEADER ERROR MESSAGES. TESTS OPERATOR
              REPLY.

IOCSNOTE    - NOTE.
                 ROUTINE FOR TYPING IOCS INDICATIONS AND ERROR MES-
              SAGES.

IOCSOPEOR   - OUTPUT END OF REEL.
                 EXIT 8 SEQUENCE. SET UP EXIT 8 ADDRESS IN THE SWITCH
              BOX IF DESIRED.

IOCSOPEORA  - OUTPUT END OF REEL AREA.
                 OUTPUT END-OF-REEL SEQUENCE. SET UP 'IEOR ' IN THE
              LABEL AREA.

IOCSOPHD    - OUTPUT HEADER.
                 RETENTION PERIOD TESTS. CHECKS IF AN OUTPUT FILE IS
              AVAILABLE FOR WRITING, OTHERWISE IT WILL CAUSE A DATE
              ERROR.

IOCSOPHDAR  - OUTPUT HEADER AB.
                 MOVE NEEDED INFORMATION FROM OLD OUTPUT HEADER TO THE
              FILE DTF AREA. MOVE THE CURRENT DATE FROM 00115 THRU
              00119 TO THE DTF AREA. ASSEMBLE A NEW OUTPUT HEADER AND
              TEST FOR THE POSSIBILITY OF AN EXIT 4 ROUTINE.

IOCSOUT     - OUT.
                 OPEN THE PENDING SWITCH SORT TABLE TO INSERT THE PEND-
              ING SWITCH ADDRESS OF THE CURRENT OPENING FILE IN CHANNEL
              PRIORITY SEQUENCE.

IOCSPAEXIT  - PRIORITY ASSIGNMENT EXIT.
                 BRANCHES TO IOCSENTA TO CONTINUE OPENING OF THE FILE.

IOCSPAHSK   - PRIORITY ASSIGNMENT HOUSEKEEPING.
                 BEGINNING OF THE PENDING SWITCH SORT ROUTINE. SETS UP
              SEQUENCE FOR ACTUAL SWITCH SORTING.

IOCSPARG    - PRIORITY ARGUMENT.
                 TWO-POSITION AREA USED FOR CONSTRUCTION OF THE CHANNEL
              PRIORITY ARGUMENT WHEN LOOKING UP AGAINST THE PENDING
              SWITCH SORT TABLE.

IOCSPENSWE  - PENDING SWITCH ENTRY.
                 RELATIVE LOCATION OF AN AREA CONTAINING THE ADDRESS OF
              THE PENDING SWITCH INSTRUCTION TEST FOR EACH FILE SCHED-
              ULER.

IOCSPRIMER  - PRIMER.
                 WRITE A TAPE MARK AFTER THE LAST BLOCK.

IOCSPRIME3  - PRIME 3.
                 WRITE THE LAST OUTPUT BLOCK.

IOCSPS      - PROGRAM STATUS.
                 FOUR MEMORY LOCATIONS USED FOR STORING THE PROGRAM
              STATUS LATCHES AFTER AN INTERRUPT. '101',G.

IOCSPSR     - PROGRAM STATUS RESTORE.
                 SEQUENCE FOR RESTORATION OF THE PROGRAM STATUS LATCHES
              PRIOR TO TURNING ON PRIORITY AND RETURNING TO THE USER'S
              PROGRAM.

IOCSRC      - HEADER RETENTION CYCLE.
                 RELATIVE LOCATION OF THE RETENTION CYCLE.

IOCSRDLIN   - READ LABEL IN.
                 READING OF THE RDLIN CARDS AND MOVING OF THE INFORMA-
              TION TO THE DTF AREA OF THE APPROPRIATE FILES.

IOCSREADRU  - READ ROUTINE.
                 LABEL READ/WRITE ROUTINE.
                 STORES THE B REGISTER FOR THE RETURN, CLEARS THE LABEL
              AREA, AND SETS THE D-MODIFIER FOR A LABEL READ.

IOCSRENTRY  - RE ENTRY.
                 RE-ENTRY POINT INTO IOCS AFTER HEADER AND TRAILER
              EXITS, AND CLEARING OF CHANNELS DUE TO AN OPEN, CLOSE,
              FEORL, OR RDLIN MACRO. STORES USER'S INDEX REGISTER 15 IN
              A HOLD AREA AND RESETS X15 TO THE CONTENTS OF DTF BOX BE-
              FORE GOING INTO IOCS.

IOCSREPLY   - REPLY.
                 FIVE-POSITION AREA FOR THE OPERATOR REPLIES DURING AN
              ERROR ROUTINE.

IOCSRSCLCT  - RESTART CHECKPOINT LOAD ON TAPE.
                 POINT OF ENTRY FROM THE CHECKPOINT RESTART PROGRAM TO
              THE PROGRAM BEING RESTARTED.

IOCSRWDB    - REWIND BRANCH.
                 CONTROL UNIT COMMAND, RELATING CHANNEL STATUS TESTS.

IOCSRWDRU   - REWIND ROUTINE.
                 MISCELLANEOUS CONTROL UNIT OPERATIONS, STORING OF THE
              RETURN ADDRESS, AND PASS CONTROL TO THE REWIND ROU-
              TINE AREA WHERE THE D-MODIFIER FOLLOWS THE BRANCH.

IOCSRWDRUA  - REWIND ROUTINE AREA.
                 PICK UP OF THE D-MODIFIER AND THE CHANNEL UNIT INFOR-
              MATION TO BE USED IN THE CONTROL UNIT OPERATION.

IOCSRWDXT   - REWIND EXIT.
                 EXIT FOR RETURN TO THE MAIN IOCS ROUTINE.

IOCSRWURU   - REWIND UNLOAD ROUTINE.
                 STORING OF THE RETURN POINT AND BRANCHING TO THE RE-
              WIND ROUTINE AREA. BRANCH IS FOLLOWED BY THE D-MODIFIER
              FOR A CONTROL UNIT OPERATION RELATING TO REWIND/UNLOAD.
```

IOCSSTLE - STALL ENTRY.
LINKAGE TO CLEAR CHANNELS OF ALL CURRENT AND PENDING OPERATIONS PRIOR TO AN OPEN, CLOSE, FEORL, OR RDLIN.

IOCSSTLEXT - STALL EXIT.
RETURN TO IOCS AFTER CLEARING THE CHANNELS AND EXECUTING EXIT ROUTINES.

IOCSSWBX - SWITCH BOX.
THIS ROUTINE RESTORES INDEX REGISTER 15 TO THE USER'S CONTENTS AFTER AN OPEN, CLOSE, FEORL, OR RDLIN, USED BY IOCS TO BRANCH TO THE USER ROUTINES IF USING EXNADDR DIOCS ENTRIES.

IOCSSWBXA - SWITCH BOX AREA.
BRANCH INSTRUCTION WHOSE OPERAND IS LOADED WITH AN EXIT ADDRESS.

IOCSTBC - TAPE BLOCK COUNT.
RELATIVE LOCATION OF THE BLOCK COUNT

IOCSTBL - TABLE.
UNITS POSITION OF THE PENDING SWITCH SORT TABLE.

IOCSTFINIT - TAPE FILE INITIALIZE.
SKELETON BRANCH INSTRUCTION MODIFIED WITH THE ADDRESS OF THE FILE INITIALIZATION ROUTINE.

IOCSTHT - TAPE HASH TOTALS.
RELATIVE LOCATION OF THE HASH TOTAL.

IOCSTPCLOS - TAPE CLOSE.
TAPE CLOSE SEQUENCE, WILL BRANCH OUT FOR PADDING POSSIBILITIES IF THE FILE BEING CLOSED IS AN OUTPUT FILE.

IOCSTRBL - TRAILER BLANKS.
BLANKS TO CLEAR THE FIRST TEN POSITIONS OF THE LABEL AREA BEFORE CONSTRUCTING A TRAILER LABEL.

IOCSTRC - TAPE RECORD COUNT.
RELATIVE LOCATION OF THE RECORD COUNT.

IOCSTRIGEN - TRIGGER END.
RELATIVE LOCATION OF A DCW CONTAINING THE ADDRESS OF THE FILE SCHEDULER BRANCH TO IOCS AFTER SIGNALING THE NECESSITY OF AN I/O OPERATION FOR THE FILE.

IOCSURANY - UNIT RECORD ANY.
SEQUENCE FOR TESTING THE STATUS LATCHES AND CONSTRUCTING ASSOCIATED CONSOLE MESSAGES TO BE TYPED OUT ON UNIT RECORD ERRORS. '20116 DCK','20143 STK', AND '20115 LLC'.

IOCSURBAN - UNIT RECORD BRANCH ANY.
BRANCH ANY AFTER THE ERROR ROUTINE RETRIES A UNIT RECORD OPERATION BEFORE RETURNING TO THE MAIN PROGRAM.

IOCSURDCK - UNIT RECORD DATA CHECK.
RETRY ROUTINE FOR DATA CHECKS, WILL ATTEMPT TWICE ON PRINTER OR PUNCH ERRORS BEFORE TYPING THE ERROR MESSAGE. WILL NOT RETRY A READER FILE.

IOCSURERR - UNIT RECORD ERROR ROUTINE.
UNIT RECORD ERROR SEQUENCE.

IOCSUREXIT - UNIT RECORD EXIT.
BRANCH FOR RETURN TO THE UNIT RECORD FILE SCHEDULER.

IOCSURNR - UNIT RECORD NOT READY.
SET UP A CONSOLE TYPEWRITER MESSAGE FOR A NOT READY INDICATION ON UNIT RECORD FILES. '10100 NR '.

IOCSUROPEN - UNIT RECORD OPEN.
CLEARS THE BLOCK COUNT ON ALL UNIT RECORD FILES DURING THE OPENING.

IOCSURPKUP - UNIT RECORD PICK UP.
INSTRUCTIONS MOVING THE UNIT RECORD COMMAND AND ITS CORRESPONDING STATUS TEST FROM THE UNIT RECORD FILE SCHEDULER TO THE ERROR ROUTINE FOR RETRY.

IOCSURQE - UNIT RECORD QUESTION ENTRY.
TEST OF A UNIT RECORD FILE SCHEDULER FOR A BEX FORM OF INSTRUCTION AS NEXT INSTRUCTION AFTER THE BRANCH TO THE ERROR ROUTINE. COULD BE CONSIDERED AS LOOKING FOR A READER FILE SCHEDULER.

IOCSURTY - UNIT RECORD TYPE.
SEQUENCE THROUGH WHICH ALL UNIT RECORD ERRORS, EXCEPT NOT READY, PASS TO HALT FOR OPERATOR INTERVENTION.

IOCSUSXA - USER SET EXIT A.
EXIT 2 SEQUENCE - MODIFY THE SWITCH BOX TO THE EXIT 2 ADDRESS IF NECESSARY, OTHERWISE WRITE A TAPE MARK.

IOCSWRITRU - WRITE ROUTINE.
STORING OF THE RETURN ADDRESS AND SETTING OF A D-MODIFIER IN THE LABEL I/O COMMAND.

IOCSWTGX - WAITING EXIT.
EXIT, AFTER FORCING CHANNEL 1 TO BE CLEARED, TO PROCEED WITH THE TYPING OF A CHANNEL 2 ERROR MESSAGE.

IOCSWTMRU - WRITE TAPE MARK ROUTINE.
STORING OF THE RETURN POINT AND BRANCHING TO THE REWIND ROUTINE AREA. BRANCH IS FOLLOWED BY A WTM D-MODIFIER FOR THE CONTROL UNIT OPERATION.

IOCSX15HD - INDEX 15 HOLD.
WILL CONTAIN THE USER'S CONTENTS OF INDEX REGISTER 15 DURING PROCESSING IN IOCS.

IOCS101 - 101.
TWO DCW'S - 10 AND 1, USED IN A COMPARE INSTRUCTION AGAINST IOCSPS FOR RESTORATION OF THE PROGRAM STATUS LATCHES PRIOR TO THE RETURN TO THE INTERRUPTED USER PROGRAM.

IOCS20 - 20.
END-OF-REEL MESSAGE. '20120 EOR'.

```
IOCS31    - 31.
              CREATION RETENTION PERIOD ERROR MESSAGE. '40131 DAT'.

IOCS32    - 32.
              INPUT HEADER CHECK ERROR MESSAGE. '30132 FIL'.

IOCS33    - 33.
              NO INPUT HEADER MESSAGE. '30133 NIH'.

IOCS34    - 34.
              INPUT TRAILER ERROR MESSAGE. '10134 TIE'.

IOCS36    - 36.
              RDLIN ERROR MESSAGE. '20136 RLN'.

IOCS44    - 44.
              '20144 WLR ' - UNIT RECORD WRONG LENGTH RECORD MESSAGE.

00000     - ACTUAL 00000.
              ACTUAL LOCATION 00000 - RELATIVE POSITIONING OF A DE-
            FINE AREA USED IN CONJUNCTION WITH THE DTF ADDRESS TO
            FIND THE NECESSARY ITEMS WITHIN ALL FILE SCHEDULERS.

100       - ACTUAL 00100.
              LOCATION OF A GROUP MARK WORD MARK USED WHEN STORING
            THE CONTENTS OF INDEX REGISTERS 13 THRU 15 INTO A SAVE
            AREA DURING AN INTERRUPT CAUSED BY AN INQUIRY OR OUT-
            QUIRY.

101       - ACTUAL 00101.
              LOCATION OF THE INTERRUPT SEQUENCE. WHEN USING THE
            PRIORITY FEATURE, THE SYSTEM AUTOMATICALLY BRANCHES TO
            THIS LOCATION WHEN THE PRIORITY ALERT LIGHT IS ON AND A
            PRIORITY REQUEST INDICATOR LATCH IS TURNED ON. THIS LO-
            CATION WILL CONTAIN A STORE B REGISTER INSTRUCTION.

108       - ACTUAL 00108.
              LOCATION OF A BRANCH EXIT PRIORITY ALERT. ANOTHER IN-
            TERRUPT CAN NOT BE SERVICED DURING IOCS.

115       - ACTUAL 00115.
              A DEFINE AREA USED AS A SPACER BETWEEN THE FIXED IN-
            TERRUPT AND THE BEGINNING OF IOCS MINIMUM AREA DEFINED
            WILL BE ENOUGH TO SKIP OVER THE PROGRAM LOAD ROUTINE.
```

Appendix F — File-Dependent Label Definitions

This section lists the labels that are generated, primarily because of DTF entries. These labels are for instructions and data areas in the file schedulers or file tables. With the label is an explanation of the routine or data area the label addresses.

```
IOCSE0000X - FILE PENDING SWITCH - CHANNEL 1.
             TAPE FILES - SYMBOLIC IDENTIFIERS FOR FILE PENDING
           SWITCHES ON CHANNEL ONE.

IOCSF0000X - FILE PENDING SWITCH - CHANNEL 2.
             TAPE FILES - SYMBOLIC IDENTIFIERS FOR FILE PENDING
           SWITCHES ON CHANNEL TWO.

IOCSXXACT  - ACTIVITY.
             ALL FILES - THE NUMERIC IDENTIFIER DESCRIBING THE KIND
           OF FILE.
               1. 1 DEFINES A ONE-AREA TAPE FILE.
               2. 2 DEFINES A TWO-AREA TAPE FILE.
               3. 3 DEFINES A CARD READER FILE.
               4. 4 DEFINES A CARD PUNCH FILE.
               5. 5 DEFINES A PRINTER FILE.
             PRIOR TO THE IOCSXXACT IS A ONE-POSITION DCW CONTAIN-
           ING THE CHANNEL PRIORITY DIGIT FOR TWO-AREA TAPE FILES.

IOCSXXBA   - BRANCH ANY.
             TAPE FILES - THE LOCATION OF THE CHANNEL STATUS TEST
           FOR THE FILE SCHEDULER.

IOCSXXBASE - BASE.
             ALL TAPE FILES - A FIVE-POSITION DCW CONTAINING MODE,
           X-CONTROL FIELD, AND STATUS TEST CHANNEL OPERATION CODE.
           FOLLOWING THIS MAY BE ANOTHER FIVE-POSITION DCW CONTAIN-
           ING THE SAME FOR AN ALTERNATE DRIVE, IF APPLICABLE.

IOCSXXBLKL - BLOCK LENGTH.
             VARIABLE BLOCKED INPUT TAPE FILE - WILL CONTAIN THE
           CONTENTS OF THE E OR F REGISTER AFTER A READ OPERATION
           FOR USE DURING THE WRONG LENGTH RECORD TEST.

IOCSXXD1   - EXIT DIGIT 1.
             ALL OUTPUT TAPE FILES - USER EXIT 1 CHECK CHARACTER.
               0 DEFINES NO EXIT 1 ADDRESS.
               1 DEFINES AN EXIT 1 ADDRESS.
             FOLLOWING THE CHECK CHARACTER WILL BE THE USER'S EXIT 1
           ADDRESS IF APPLICABLE. OTHERWISE, A FIVE-POSITION
           BLANK DCW.

IOCSXXD2   - EXIT DIGIT 2.
             ALL OUTPUT TAPE FILES - USER EXIT 2 CHECK CHARACTER.
               0 DEFINES NO EXIT 2 ADDRESS.
               1 DEFINES AN EXIT 2 ADDRESS.
             FOLLOWING THE CHECK CHARACTER WILL BE THE USER'S EXIT 2
           ADDRESS IF APPLICABLE. OTHERWISE, A FIVE-POSITION
           BLANK DCW.

IOCSXXD3   - EXIT DIGIT 3.
             ALL OUTPUT TAPE FILES - USER EXIT 3 CHECK CHARACTER.
               0 DEFINES NO EXIT 3 ADDRESS.
               1 DEFINES AN EXIT 3 ADDRESS.
             FOLLOWING THE CHECK CHARACTER WILL BE THE USER'S EXIT 3
           ADDRESS IF APPLICABLE. OTHERWISE, A FIVE-POSITION
           BLANK DCW.

IOCSXXD4   - EXIT DIGIT 4.
             ALL OUTPUT TAPE FILES - USER EXIT 4 CHECK CHARACTER.
               0 DEFINES NO EXIT 4 ADDRESS.
               1 DEFINES AN EXIT 4 ADDRESS.
             FOLLOWING THE CHECK CHARACTER WILL BE THE USER'S EXIT 4
           ADDRESS IF APPLICABLE. OTHERWISE, A FIVE-POSITION
           BLANK DCW.

IOCSXXD5   - EXIT DIGIT 5.
             ALL OUTPUT TAPE FILES - USER EXIT 5 CHECK CHARACTER.
               0 DEFINES NO EXIT 5 ADDRESS.
               1 DEFINES AN EXIT 5 ADDRESS.
             FOLLOWING THE CHECK CHARACTER WILL BE THE USER'S EXIT 5
           ADDRESS IF APPLICABLE. OTHERWISE, A FIVE-POSITION
           BLANK DCW.

IOCSXXD6   - EXIT DIGIT 6.
             ALL INPUT TAPE FILES - USER EXIT 6 CHECK CHARACTER.
               0 DEFINES NO EXIT 6 ADDRESS.
               1 DEFINES AN EXIT 6 ADDRESS.
             FOLLOWING THE CHECK CHARACTER WILL BE THE USER'S EXIT 6
           ADDRESS IF APPLICABLE. OTHERWISE, A FIVE-POSITION
           BLANK DCW.

IOCSXXD7   - EXIT DIGIT 7.
             ALL INPUT TAPE FILES - USER EXIT 7 CHECK CHARACTER.
               0 DEFINES NO EXIT 7 ADDRESS.
               1 DEFINES AN EXIT 7 ADDRESS.
             FOLLOWING THE CHECK CHARACTER WILL BE THE USER'S EXIT 7
           ADDRESS IF APPLICABLE. OTHERWISE, A FIVE-POSITION
           BLANK DCW.

IOCSXXD8   - EXIT DIGIT 8.
             ALL OUTPUT TAPE FILES - USER EXIT 8 CHECK CHARACTER.
               0 DEFINES NO EXIT 8 ADDRESS.
               1 DEFINES AN EXIT 8 ADDRESS.
             FOLLOWING THE CHECK CHARACTER WILL BE THE USER'S EXIT 8
           ADDRESS IF APPLICABLE. OTHERWISE, A FIVE-POSITION
           BLANK DCW.

IOCSXXEMTY - EMPTY.
             CARD READER AND INPUT TAPE FILES - THE ENTRY POINT TO
           THE FILE SCHEDULER FROM A GET MACRO-INSTRUCTION WHEN A
           READ OPERATION IS TO BE PERFORMED.

IOCSXXENDA - END A.
             FIXED BLOCKED FILES - ADDRESS OF THE LAST RECORD MARK
           OF I/O AREA A. THIS IS INSERTED INTO IOCSXXENDD WHEN IN-
           FORMATION IN THE AREA IS READY TO BE PROCESSED.

IOCSXXENDB - END B.
             FIXED BLOCKED FILES - ADDRESS OF THE LAST RECORD MARK
           OF I/O AREA B. THIS IS INSERTED INTO IOCSXXENDD WHEN IN-
           FORMATION IN THE AREA IS READY TO BE PROCESSED.
```

IOCSXXENDD - END DIGIT.
BLOCKED TAPE FILES - LOCATION OF THE CURRENT INFORMATION AREA RECORD MARK. USED BY THE GET TO COMPARE AGAINST IOCSXXSAVE FOR INITIATING A BRANCH TO THE FILE SCHEDULER ON FIXED BLOCKED FILES AND VARIABLE BLOCKED INPUT FILES. ON VARIABLE BLOCKED OUTPUT FILES, THIS FIELD IS COMPARED AGAINST IOCSXXRLAC BEFORE INITIATING THE BRANCH.

IOCSXXENDI - END INITIALIZER.
BLOCKED TAPE FILES - ZERO ADDED TO IOCSXXSAVE DURING INITIALIZATION TO FORCE A BRANCH TO THE FILE SCHEDULER ON THE FIRST GET TO AN INPUT FILE. ALTHOUGH DEFINED FOR OUTPUT FILES, IT IS NOT USED.

IOCSXXEOF - END OF FILE.
CARD READER AND ALL INPUT TAPE FILES - WILL CONTAIN THE USER'S END-OF-FILE ROUTINE ADDRESS.

IOCSXXEXIT - EXIT.
ALL FILES EXCEPT PASSIVE TAPE FILES - BRANCH BACK TO THE USER'S ROUTINE.

IOCSXXFSCK - FILE SERIAL CHECK.
ALL TAPE FILES - THE CHECK CHARACTER FOR FILE SERIAL. FOR INPUT FILES, WILL CHECK THE FILE SERIAL NUMBER AGAINST IOCSXXHFS. OUTPUT FILES, IOCS WILL READ LABELS AND RETAIN THE FILE SERIAL DURING THE CONSTRUCTION OF A NEW LABEL AND INSERT THE ORIGINAL FILE SERIAL NUMBER INTO THE NEW LABEL PRIOR TO WRITING IT.
0 DEFINES CHECKING OF THE FILE SERIAL NUMBER.
1 DEFINES NO CHECKING OF THE FILE SERIAL NUMBER.

IOCSXXFULL - FULL.
PRINTER, PUNCH, AND OUTPUT TAPE FILES - THE ENTRY POINT INTO THE FILE SCHEDULER FROM A PUT INSTRUCTION WHEN A WRITE OPERATION IS TO BE PERFORMED.

IOCSXXHFS - HOLD FILE SERIAL.
ALL TAPE FILES - THE HOLD AREA FOR THE FILE SERIAL NUMBER WILL CONTAIN THE SERIAL NUMBER TO CHECK AGAINST LABELS ON INPUT FILES. ON OUTPUT FILES, THE FILE SERIAL NUMBER IS MOVED HERE AND IS USED IN THE CONSTRUCTION OF THE NEW HEADER LABEL. FOLLOWING THIS HOLD AREA WILL BE THE REMAINING INFORMATION NECESSARY TO CONSTRUCT OR CHECK THE HEADER LABEL FOR THE FILE.

IOCSXXINIT - INITIALIZATION.
TAPE FILES - SEQUENCE TO INITIALIZE FILE I/O COMMANDS AND FORCE A DOUBLE READ ON TWO-AREA FILES DURING THE FIRST GET COMMAND TO THAT FILE. INDEX REGISTERS ARE INITIALIZED ON OUTPUT FILES.

IOCSXXIOA - I/O A.
ALL FILES EXCEPT PASSIVE TAPE FILES - THE I/O COMMAND THAT WILL CAUSE READING TO OR WRITING FROM I/O AREA A. IN UNIT RECORD FILE SCHEDULERS, IT WILL BE THE I/O COMMAND FOR THAT SCHEDULER.

IOCSXXIOAR - I/O AREA.
UNIT RECORD FILES - AN EQUATE TO THE AREA SPECIFIED BY THE IOAREA DTF ENTRY.

IOCSXXIOB - I/O B.
TAPE FILES - THE I/O COMMAND THAT WILL CAUSE READING TO OR WRITING FROM I/O AREA B.

IOCSXXPA - PRIMARY AREA.
TAPE FILES - TESTS FOR THE LAST I/O AREA USED AND SETS UP NEXT I/O COMMAND TO USE I/O AREA A.

IOCSXXPADS - PADDING SEQUENCE.
BLOCKED OUTPUT FILES - FIXED, WRITING OF THE LAST OUTPUT BLOCK. TESTS FOR THE NECESSITY OF PADDING, SAVES AND SETS THE CONTENTS OF INDEX REGISTER 15 WITH THE ADDRESS OF THE FIRST LOCATION TO START PADDING. VARIABLE, WRITING OF THE LAST BLOCK ONLY.

IOCSXXPFOR - PADDING FORCE.
FIXED BLOCKED OUTPUT FILES - RESTORE INDEX REGISTER 15 AND BRANCH TO WRITE THE PADDING RECORD.

IOCSXXPLB1 - PADDING LAST BLOCK 1.
FIXED BLOCKED OUTPUT FILES - UPDATE INDEX REGISTER 15 TO PASS THE PADDING OF A RECORD MARK POSITION AND BRANCH BACK TO THE PADDING ROUTINE.

IOCSXXPLB2 - PADDING LAST BLOCK 2.
FIXED BLOCKED OUTPUT FILES - TEST FOR RECORD MARKS WITHIN THE AREA TO BE PADDED. TESTS FOR END OF PADDING ROUTINE.

IOCSXXPRIM - PRIMARY.
INPUT TAPE FILES - SETS UP TO FORCE TWO CONSECUTIVE READS ON THE FIRST GET INSTRUCTION TO THE FILE.

IOCSXXPSVE - PADDING SAVE.
FIXED BLOCKED OUTPUT FILES - A FIVE-POSITION DCW FOR STORING THE CONTENTS OF INDEX REGISTER 15 DURING THE PADDING OF THE LAST BLOCK.

IOCSXXPTRC - PADDING - TAPE RECORD COUNT.
FIXED BLOCKED OUTPUT FILES - ACCUMULATE HASH TOTALS.

IOCSXXRCLN - RECORD LENGTH.
FIXED BLOCKED FILES - CONTAINS THE RECORD LENGTH FOR THIS FILE.

IOCSXXRLAC - RECORD LENGTH AT CURRENT.
VARIABLE BLOCKED OUTPUT TAPE FILE - WILL CONTAIN THE ADDRESS OF THE LAST LOCATION USED FOR STORING DATA. IT IS USED TO DETERMINE WHEN THE I/O AREA HAS BEEN FILLED.

IOCSXXSA - SECONDARY AREA.
TAPE FILES - SETS UP NEXT I/O COMMAND TO USE I/O AREA B.

IOCSXXSAVE - SAVE.
TWO-AREA TAPE FILES- WILL CONTAIN THE LOCATION OF THE AREA OR SEGMENT OF THE AREA THAT HOLDS CURRENT INFORMATION. THE GET AND PUT MACROS UPDATE THIS AREA UNTIL THE COMPLETE AREA HAS BEEN USED. AT THIS TIME, A GET OR PUT MACRO WILL CAUSE THE NEXT AREA TO BECOME AVAILABLE AND CONDITION THE FILE SCHEDULER TO INITIATE ANOTHER I/O OPERATION.

IOCSXXSFS - SCHEDULER FORCE SEQUENCE.
FORCES THE CHANNEL TO CLEAR, AND BACKSPACES THE TAPE FOR A RETRY

IOCSXXSFX - SCHEDULER FORCE EXIT.
ALL FILES EXCEPT PASSIVE TAPE FILES - BRANCHING TO THE CHANNEL SCHEDULER WHEN FORCING THE PREVIOUS OPERATION TO BE STARTED OR COMPLETED. WILL FORCE THE CHANNEL TO BE CLEARED.

IOCSXXSVRL - SINGLE VARIABLE RECORD LENGTH.
VARIABLE BLOCKED OUTPUT TAPE FILES - DURING A PUT OPERATION. THIS WILL BE LOADED WITH THE CURRENT RECORD LENGTH. THIS IS ADDED TO IOCSXXRLAC TO CHECK IF THE OUTPUT AREA HAS BEEN FILLED.

IOCSXXTBC - TAPE BLOCK COUNT.
ALL FILES - DTF WORK AREA FOR BLOCK COUNT.

IOCSXXTFLB - TAPE FILE LABEL.
ALL TAPE FILES - THE CHECK CHARACTER DEFINING TYPE OF TAPE LABELS.
0 DEFINES STANDARD LABELS.
1 DEFINES AN UNLABELLED FILE.
2 DEFINES NON STANDARD LABELS.

IOCSXXTFL1 - TAPE FILE LABEL 1.
ALL TAPE FILES - FILE TYPE CHECK CHARACTER.
0 DEFINES OUTPUT OR PASSIVE TAPE FILES.
1 DEFINES INPUT FILES.

IOCSXXTFL2 - TAPE FILE LABEL 2.
ALL TAPE FILES - ALTERNATE DRIVE CHECK CHARACTER.
0 DEFINES NO ALTERNATE DRIVE.
1 DEFINES AN ALTERNATE DRIVE.

IOCSXXTFL3 - TAPE FILE LABEL 3.
ALL TAPE FILES - CHECK LABEL CHECK CHARACTER.
0 DEFINES A CHECK OF THE COMPLETE LABEL.
1 DEFINES NO LABEL CHECKING.
2 DEFINES IDENTITY CHECKING ONLY.

IOCSXXTFL4 - TAPE FILE LABEL 4.
ALL TAPE FILES - TAPE MARK AFTER HEADER CHECK CHARACTER.
0 DEFINES NO TAPE MARK.
1 DEFINES A TAPE MARK.

IOCSXXTFL5 - TAPE FILE LABEL 5.
ALL TAPE FILES - REWIND OPTION CHECK CHARACTER.
0 DEFINES NO REWIND.
1 DEFINES REWIND ONLY.
2 DEFINES REWIND UNLOAD.

IOCSXXTHT - TAPE HASH TOTAL.
ALL TAPE FILES - THE DTF WORK AREA FOR ACCUMULATING HASH TOTALS FOR THE FILE.

IOCSXXTRC - TAPE RECORD COUNT.
ALL TAPE FILES - DTF WORK AREA FOR RECORD COUNT. RECORD COUNT IS UNOBTAINABLE FOR UNBLOCKED FILES.

IOCSXXTRIG - TRIGGER.
ALL FILES EXCEPT PASSIVE TAPE FILES - BRANCH BACK TO IOCSENTRY AFTER THE SETTING OF THE PENDING SWITCH. INITIALLY SET TO FORCE TWO READS FOR AN INPUT FILE ON THE FIRST GET OPERATION. ON VARIABLE BLOCKED INPUT FILES, IT CHECKS THE LAST I/O OPERATION FOR A WRONG-LENGTH RECORD. UNIT RECORD FILES BRANCH TO IOCSENTRY.

IOCSXXVBCA - VARIABLE BLOCK COUNT A.
FIXED BLOCKED OUTPUT AND VARIABLE BLOCKED FILES - FOR A FIXED BLOCKED OUTPUT FILE, THIS CONTAINS THE ADDRESS OF THE GROUP MARK WORD MARK LOCATION FOR I/O AREA A. WILL BE USED DURING THE PADDING ROUTINE. FOR VARIABLE BLOCKED FILES, WILL CONTAIN THE UNITS POSITION OF THE BLOCK CHARACTER COUNT FOR I/O AREA A.

IOCSXXVBCB - VARIABLE BLOCK COUNT B.
FIXED BLOCKED OUTPUT AND VARIABLE BLOCKED TAPE FILES - WILL CONTAIN THE LOCATION OF THE GROUP MARK WORD MARK POSITION OF I/O AREA B. WILL BE USED DURING THE PADDING ROUTINE FOR A FIXED BLOCKED OUTPUT FILE. WILL CONTAIN THE ADDRESS OF THE BLOCK CHARACTER COUNT UNITS POSITION FOR VARIABLE BLOCKED FILES.

IOCSXXVBSA - VARIABLE BLOCK SINGLE A.
VARIABLE BLOCKED TAPE FILES - AN ADCON OF THE UNITS POSITION OF I/O AREA A, MOVED INTO IOCSXXENDD TO CONTROL THE AMOUNT OF THE I/O AREA REMAINING TO BE USED BY A GET OR PUT INSTRUCTION.

IOCSXXVBSB - VARIABLE BLOCK SINGLE B.
VARIABLE BLOCKED TAPE FILES - AN ADCON OF THE UNITS POSITION OF I/O AREA B, MOVED TO IOCSXXENDD TO CONTROL THE AMOUNT OF THE I/O AREA REMAINING TO BE USED BY A GET OR PUT INSTRUCTION.

IOCSXXWLR - WRONG-LENGTH RECORD.
VARIABLE BLOCKED INPUT FILES - WRONG-LENGTH RECORD SEQUENCE. UPDATES THE WRONG-LENGTH RECORD COUNT AND TESTS FOR TEN RETRIES. IF SO, WILL BRANCH TO IOCSXXWLX.

IOCSXXWLRC - WRONG-LENGTH RECORD COUNT.
VARIABLE BLOCKED INPUT TAPE FILES - A COUNT OF THE NUMBER OF TIMES THE WRONG-LENGTH RECORD ROUTINE WAS ENTERED.

IOCSXXWLRX - WRONG LENGTH RECORD EXIT.
VARIABLE BLOCKED INPUT - IF THE BLOCK CHARACTER COUNT DOES NOT AGREE WITH THE NUMBER OF CHARACTERS READ, THIS WILL BRANCH TO THE WRONG-LENGTH RECORD ROUTINE.

IOCSXXWLX - WRONG LENGTH EXIT.
VARIABLE BLOCKED INPUT FILES - BRANCH TO THE USER'S WRONG-LENGTH RECORD ROUTINE ADDRESS.

IOCSXXWORK - WORK.
ALL FILES EXCEPT PASSIVE TAPE FILES - THE ADDRESS OF THE WORK AREA ASSIGNED TO THIS FILE.

IOCSXXWTG - WAITING.
TAPE FILES - TESTS THE PENDING SWITCH FOR PENDING OPERATIONS. IF NOT, CONTINUE TO READ OR WRITE OPERATION.

EVALUATION SHEET
IBM 1410 INPUT/OUTPUT CONTROL SYSTEM

PROGRAMMING SYSTEMS ANALYSIS GUIDE. FORM C28-0541-1

FROM

NAME _____

OFFICE NO. _____

FOLD

CHECK ONE OF THE COMMENTS AND EXPLAIN IN THE SPACE PROVIDED

FOLD

☐ SUGGESTED ADDITION (PAGE , TIMING CHART, DRAWING, PROCEDURE, ETC.)

☐ SUGGESTED DELETION (PAGE)

☐ ERROR (PAGE)

EXPLANATION

CUT ALONG LINE

FOLD

FOLD

NO POSTAGE NECESSARY IF MAILED IN U.S.A.
FOLD ON TWO LINES, STAPLE, AND MAIL

STAPLE

STAPLE

FOLD

FOLD

FIRST CLASS
PERMIT NO. 81

POUGHKEEPSIE, N. Y.

BUSINESS REPLY MAIL
NO POSTAGE STAMP NECESSARY IF MAILED IN U. S. A.

POSTAGE WILL BE PAID BY

IBM CORPORATION
P.O. BOX 390
POUGHKEEPSIE, N.Y. 12602

ATTN: PROGRAMMING SYSTEMS, DEPARTMENT D91

CUT ALONG LINE

FOLD

FOLD

2/64:5M-VO-10(

STAPLE

STAPLE

Printed in U.S.A. C28-0541-1

IBM

International Business Machines Corporation
Data Processing Division
112 East Post Road, White Plains, N.Y. 10601