Reference Manual IBM 709/7090 Input/Output Control System

# MAJOR REVISION (January 1962)

This manual, C28-6100-2, is a major revision of the previous edition, C28-6100-1, and supersedes that edition.

Second Printing

Address comments regarding this publication to Programming Systems Publications, IBM Corporation, P.O. Box 390, Poughkeepsie, N. Y.

## **Preface**

The IBM 709/7090 Input/Output Control System (10CS) is designed to relieve programmers of the necessity of writing input and output routines. When using the system, a programmer need be concerned only with the content of his information files, and not with the physical processes by which information is read and written. He can, if he so chooses, think of each tape reel as consisting of a continuous string of words, and each tape drive as having the ability to start and stop at any point. During processing, 10CS automatically handles label checking and preparation, blocking and unblocking of data words, and overlapping of processing with input and output. Provision is also made for error detection and correction, checkpoint and restart procedures, tape switching, and tape assignment at execution time.

These automatic features depend upon the use of the Data Channel Trap; therefore, rocs cannot be used with 709 systems unless they are equipped with this device. In addition to the Data Channel Trap, rocs is a participating system under IBSYS, Basic Monitor, and must be used with it.

Note that any program which uses 10cs to control input/output functions must use the system for all its 1/0 functions and must not use any input/output routines other than those of 10cs and/or Basic Monitor.

This manual presents a description of the features of rocs and includes all details necessary for using the system. The material covered falls into three categories: programming aspects of the system (Part I), a sample problem (Part II), and operation of the system (Part III). It is assumed throughout that the reader has a thorough knowledge of the 709 or 7090, and of symbolic programming techniques.

## **Contents**

INTRODUCTION	1
File Description	1
Buffer Definition	1
File Initialization	1
File Processing	2
File Closing	2
PART I: PROGRAMMING USING IOCS	3
Section 1: The Data File	3
Section 2: Buffer and Buffer Pool Definition	4
Section 3: File Initialization	6
Attachment of Immediate Files	6
Attachment of Reserve Files	7
Attachment of Internal Files	7
File List	7
Opening Reserve Files	9
Section 4: File Processing	10
IOCS Commands	10
Transmitting Commands	
Transmitting Commands	11
Buffer Control: IORP, IORT	11
Special Count Control: IOSP, IOST	12
Buffer Control: IORP, IORT	12
Examples	12
Examples  Non-transmitting Commands  Count Control: IOCPN, IOCTN, IOCDN	13
Count Control: IOCPN, IOCTN, IOCDN	13
Buffer Control: IORPN, IORTN	14
Buffer Control: IORPN, IORTN	14
Examples	14
IOCS Routines	15
READ	15
WRITE	16
History Records	17
Transfer of Data	18
COPY	18
STASH	18
Other System Routines	18
CLOSE	18
Non-data Functions	19
WEF	19
REW	20
BSR	20
BSF	20
Section 5: Labels and Labeled File Procedures	21 21
Format of Labeled Tapes	21
Standard Label Formats	21
Header Label	$\frac{21}{22}$
Trailer Label	23
Labeling and Label Checking	23
Input File Header Label	23
Additional Header Records on Input File	24
Input File Trailer Label	24
Output File Header Label	24
Output File Trailer Label	24
Non-standard Labels	24
Section 6: Unlabeled File Procedures	26
Single-reel Unlabeled Files	26
Multi-reel Unlabeled Files	26
Multi-file Reels	26
Section 7: Use of the Reserve Group Option	27
Deferred Opening of a File	27
Using a File for both Input and Output	27
Using More than one Buffer for a File	28

Section 8: Use of the Internal Group Opening an Internal File Closing an Internal File Processing an Internal File READ WRITE STASH Section 9: Block Sequence Numbers and Check Sums Section 10: Checkpoint and Restart Section 11: Density and Mode Conventions Density Mode Section 12: Printing Messages On-Line Section 13: Program Preparation Storage Conventions References to System Subroutines Completed Program Return Point File Control Blocks File Block Unit Control Blocks Configurations Options Control Cards JOB Card FILE Card DATE Card LOAD Card RESTART Card IBSYS Card IID Card	30 31 31 31 31 33 34 35 35 36 38 38 40 41 42 42 42 42 42 42 42
Control Card Formats	42 42
FILE Card DATE Card LOAD Card RESTART Card IBSYS Card ID Card Program Input Program Loading	43 45 45 45 46 46 46
PART II: EXAMPLE OF THE USE OF IOCS	47 47
Output Processing Programming the Input/Output File Specifications Buffer Pools Initialization Processing Master Files Processing Transaction Files End of File: Transaction File End of File: Old Master File Variable Length Records	48 48 48 49 50 51 51 52 53
PART III: IOCS OPERATION  Section 1: Control Sense Switches  Input Control Restart Control Checkpoint Control Label Control Error History Control Section 2: System Operating Notes Start Procedure System Operation Preprocessor Error List Preprocessor Error Messages	55 55 55 55 55 55 56 56 56 57 57

Preprocessor File List	57
IOĈS System Messages and Halts	58
Tape Mounting and Removal Instructions	59
Normal Label Messages	<b>5</b> 9
	59
	60
Label Error Messages	60
	61
File Processing Error Messages	61
Tape Error Messages	62
Configuration Error Message	62
Section 3: Checkpoint and Restart Procedures	63
Operator Initiated Checkpoint	63
	63
Messages During Restart	63
Restart Error Messages	64
ADDRIVE A STATE COMMENT DE COM HODICE	~ <del>-</del>
	65
	69
	69
	<b>6</b> 9
	69
Use of Copy When "Reading" with Transmitting Commands	69
Re-use of a Buffer Pool	70
Joining Pools	70
Actions of IOCS Routines Under Abnormal Conditions	70
IOCS Command Execution Tables	72
CIOCCADY	75
GLOSSARYINDEX	75 70
INDEX	79

## Introduction

The IOCS system is composed of two basic parts: one, the Preprocessor, which is the initialization routine of the system; and two, IOCS itself, which controls input/output during the execution of the object program.

The Preprocessor requires a set of control cards which describe the files rocs is to process. It loads rocs into storage, sets up the desired file specifications in the format required by rocs using control card information, and initiates loading of the object program. Use of the Preprocessor allows variation in the file description and the system procedure for each computer run. Hence, many different kinds of files can be processed by the same program with no change except in the file descriptions furnished to the Preprocessor.

rocs is a set of closed subroutines that automatically overlap input/output and processing — the process normally referred to as *buffering*. In addition, rocs provides a scheme for "reading" and "writing" groups of data (any number of machine words) from or onto input/output files, irrespective of the physical arrangement of data in the file.

The use of iocs is divided into five distinct functions, as described below, all of which are necessary in every program using iocs. Each function is provided for by iocs routines or procedures.

### File Description

Information necessary for iocs processing of a file must be provided. This includes a specification of the i/o unit(s), the file type (input or output), file mode, density and labeling conventions. System options such as sequence checking of physical tape blocks, calculation and verification of check sums, and multiple reel file handling can also be specified. File description is accomplished through the use of the Preprocessor file cards (see page 43).

iocs requires twelve words — a File Control Block — for control information pertaining to each file to be processed using the system. All references to a particular file in an object program are made by referring to the first word of the File Control Block of that file.

#### **Buffer Definition**

I/O areas, referred to as *buffers*, must be defined for use by IOCS. The efficiency which can be obtained in the overlapping of processing and input/output is directly related to the assignment of buffer areas and the size of each individual buffer.

#### File Initialization

rocs must be provided with an indication to start reading ahead on an input file, or to prepare to overlap writing on an output file. Label checking and preparation is also accomplished. The initialization process consists of two steps:

- a. Attachment of the file to previously defined buffer area(s). 10cs differs from most buffering systems in that the file is attached to buffers rather than buffers being attached to the file. In fact, several files will normally be attached to the same group of buffer areas, called a Buffer Pool. Because of this feature, a buffer within a pool can be used at different times by different files. 10cs adjusts the number of buffers used for each file according to the relative volume of data in the file and the frequency of use.
- b. Opening of a file initiates input/output functions for the file.

## **File Processing**

Processing is accomplished by the use of the iocs routines READ, WRITE, and COPY. With the exception of COPY (which is used to transfer information from an input file to an output file), these routines are directed by "commands" which form part of the calling sequences for the routines. iocs commands have the same format as 709/7090 i/o commands, and their interpretation is quite similar. Each command "reads" or "writes" some multiple of machine words from or onto the specified file. Optionally, the data may be either transmitted to a working storage or located within the buffer area(s) with which the file is associated. Transmission of data between the buffers and the input/output units is controlled automatically by iocs.

#### File Closing

Closing a file provides the indication to locs that the usage of a particular file is to be terminated. For unlabeled files, the WEF and REW routines may be used to temporarily suspend operation; normal operation will be continued when the file is again referred to by a READ or WRITE. All I/O activity required for a file between the opening and closing of the file, is overlapped with the execution of locs and the *object* program.

These functions are described more fully in the following pages.

## Part I:

## **Programming Using IOCS**

## Section 1—The Data File

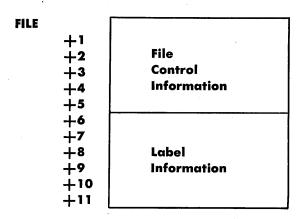
A Data File is a collection of information stored on some external data recording device: rocs is able to handle tape, card, and printed files. The information so stored has some logical grouping, each group being called a logical record. A logical record may consist of one word, several words, an entire block (one physical record of the external storage media, i.e., a card, a tape record, or a line of print), more than one block (e.g., all of one block and part of another), or several blocks. The form of each logical record may be the same as, or different from, the preceding one. The definition of the logical record is expressed implicitly in the user's choice of rocs commands and not explicitly by any definition to the system.

In this manual, the term *File* has three connotations: first, a logical collection of data; second, an external device which holds this data; and third, the symbolic location of the first word in the File Control Block. In general, when the term File is used, the explicit reference intended is the symbolic location of the File Control Block. Appendix A contains a detailed description of the File Control Block; however, the following explanation will suffice.

Twelve locations must be reserved for each locs file, these being reserved by BSS instructions such as:

MASFIL	BSS	12	MASTER FILE
DETFIL	BSS	12	DETAIL FILE
	•		
	•		

Of these, the first six are used for control information of the file, and the last six for labeling information. In no case should any information be loaded into these locations by the object program, since they are used by the Preprocessor to store information generated from file cards.



## Section 2-Buffer and Buffer Pool Definition

A buffer is an area used for intermediate storage of input/output data. Buffers are used by iocs to hold an input block until it can be processed and to hold an output block until it can be written. It is analogous to the in-out box found on most office desks. The in tray is an input buffer, a place to temporarily hold information yet to be processed. The out tray is then an output buffer; it temporarily holds information already processed and waiting to be filed.

A buffer pool is a group of buffers connected so that a file using the pool can use any of the available buffers for holding information. Of course, two files cannot use the same buffer at the same time. Normally, the size of each buffer in a pool will correspond to the size of the blocks in the files using that pool.

Every buffer pool has two control words which serve to control usage of the buffers within the pool. In addition, the first two words of each buffer are control words; one is used to keep a record of the status of information within the buffer, and the other serves as a machine 1/0 command when the contents of the buffer are to be written, or a block is to be read into the buffer. Thus, if N is the maximum number of data words a buffer is to hold, and if the pool is to consist of M buffers, the programmer must reserve M(N+2)+2 cells. For example, to reserve space for a pool of eight 30-word buffers, the following symbolic instruction could be used:

POOL BSS 8\*30+8\*2+2

The structure of a buffer pool is set up by:

* FOR RE	MARKS		
LOCATION	OPERATION	ADDRESS, TAG, DECREMENT/COUNT	C(
1 2 6	7 8 14	4 15 16	\
11	TSX	DEFINE	)
	P.Z.E.		ائب
	P.Z.E.	M, , , , N	

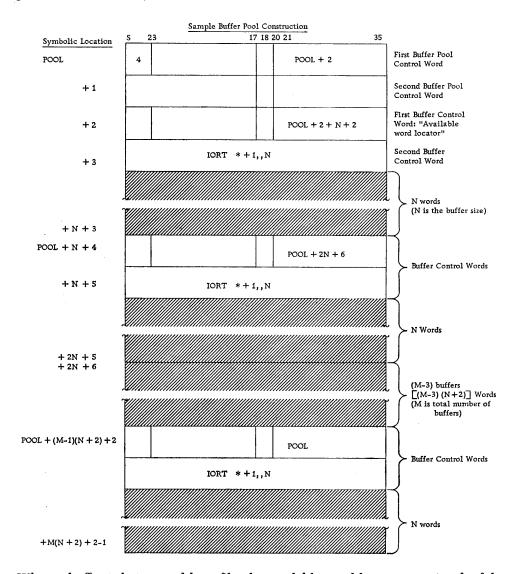
This calling sequence defines the area beginning with symbolic location POOL as a pool of M buffers, each of effective size N, with the first buffer pool control word having symbolic location POOL. All further references to this pool are specified by referring to the symbolic location POOL.

The use of the above sequence requires the area to be a single block. A pool consisting of buffers scattered within storage can be defined by the use of

	LOCATION		OPERATION		ADDRESS, TAG, DECREMENT/COUNT	C
_	2 6	7	8	4	16	
			T.S.X.	<u> </u>	J.O.I.N., 14	
			PZE		P.O.O.L 1,P.O.Q L 2	
				-		

This sequence causes previously defined pools at symbolic locations POOL1 and POOL2, to be considered as one pool; the buffers of each must be the same size. POOL1 is the symbolic location of the first control word of the enlarged pool. POOL2 cannot subsequently appear in a calling sequence for any IOCS routine, since it is no longer the location of a buffer pool control word.

Either of these two defining sequences may be included at any point in a program. However, if any of the buffers involved contain data, this data will be lost.



When a buffer is being used by a file, the available word locator contains the following: bits 21-35, the location of the next available word; bits 3-17, the number of available words. When a buffer is not in use by any file, the first word contains the address of another buffer in the pool which is not in use (this is the *Chain Address*.) If there are no more available buffers in the pool, the Chain Address will be the location of the first Buffer Pool Control Word.

#### Section 3—File Initialization

Initialization of files consists of two steps: one, assignment of one or more files to use a given buffer pool; and two, opening those files. As a convenience, certain files may be "attached" and "opened" in one step; these files are called *Immediate files*. There are two other classes of files, *Reserve* and *Internal*, which must be attached and opened in separate steps.

Every file must be attached and opened before it can be referenced by any other rocs routine. Every file that is to use a given buffer pool must be specified in one calling sequence to the ATTACH routine. A group of files to use a certain buffer pool is attached by the calling sequence:

* FOR REM	ARKS	
LOCATION	OPERATION	ADDRESS, TAG, DECREMENT/COUNT CC
1 2 6 7	7 8 14	15 16
	T <sub>1</sub> S <sub>1</sub> X <sub>1</sub>	A.T.T.A.CH., 4
	$P_iZ_iE_i$	P.O.O.L.
	P.Z.E.	LI S,T,,,,FI LECT

#### where

LIST is the symbolic location of the list of files (see page 7),

FILECT is the number of files (a decimal number) in the list, and
POOL is the symbolic location of the desired buffer pool.

This routine establishes a connection between the files and the buffer pool.

#### Attachment of Immediate Files

An Immediate file is a regular input/output file which is to be processed only once during the course of the program. It is automatically rewound and opened when attached. The number of Immediate files that can be specified in the list is dependent upon their type and the number of buffers in the pool. Three types of files can be designated to rocs. The three permissible types and their minimum buffer requirements are as follows:

- 1. Input files (I) each input file requires at least one buffer at all times.
- 2. Partial block output files (P) this type of file permits writing of any number of machine words regardless of buffer size. Each type P file requires at least one buffer.
- 3. Total block output file (T) this type of output file requires writing of exact number of physical blocks each time it is referred to by a WRITE sequence. As this type of file does not withhold a buffer from the pool except when the file is in use, all files of this type could use the same buffer at different times. Thus, all type T files which use one pool require only one buffer.

A simple example will help clarify the determination of buffer requirements for a group of files. Suppose that there are two input files, three partial block output files, and four total block output files, that these are all Immediate files, and that all are to use the same buffer pool. Then this pool must have at least six buffers: two for the type I files, three for the type P files, and one for all four type T files. Naturally, the pool may include additional buffers to take advantage of the overlapping possibilities; however, the pool must have at least six buffers for the attachment of those files to be allowed.

#### **Attachment of Reserve Files**

A Reserve file is a regular input/output file for which a more complex processing is required than is available with an Immediate file. It is neither rewound nor opened when attached. Reserve files are specified in groups, although one Reserve Group may consist of only one file. For each Reserve Group in a file list, there must be one control word which specifies the number of buffers to be reserved for the group and the maximum number of files in the group that are going to be open at the same time. The number of buffers specified for a Reserve Group must not be less than that obtained by calculating the buffer requirements of the Reserve files which are to be opened simultaneously. The buffer requirements for Reserve files are determined in the same way as for Immediate files, except that output files require one buffer each.

As an example, suppose that a group of nine files (two input files, three partial block output files, and four total block output files) are to use one pool and that it is known that only one of the input files and one of the type P output files will be open at the same time. If these files were all specified as Immediate files as shown before, they would require at least six buffers. However, the two input files and the three P output files may be formed into a Reserve Group, with the control word specifying that the number of files open simultaneously is two (one input, one type P) and that the buffer requirement is two (one for each). Hence, the minimum number of buffers required is now only three: two for the Reserve Group and one for the four type T (Immediate) files.

A more detailed discussion of Reserve files and their usage is given in Section 7.

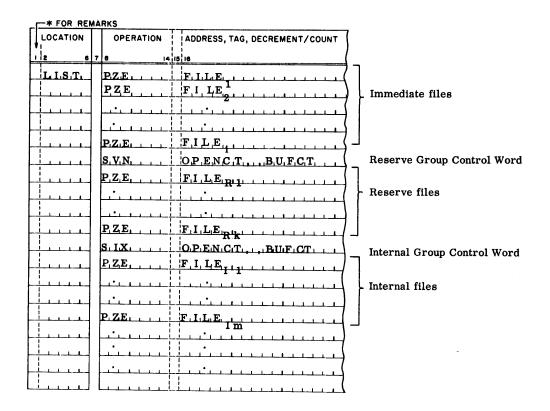
#### **Attachment of Internal Files**

An Internal file is a file which is kept entirely in core storage and has no 1/0 unit associated with it. From a programming point of view, however, Internal files have much the same characteristics as Reserve files; and like them, are specified in groups in the file list (using an Internal Group control word). The minimum buffer requirements for the group are determined in the same way as for a Reserve Group.

A discussion of Internal files and their usage is given in Section 8.

## File List

The list of files which is to be attached to a buffer pool must be of the following form, all Immediate files preceding all Reserve and Internal Groups:



#### where

FILE, are Immediate files,

FILE<sub>Ri</sub> are Reserve files,

FILE11 are Internal files,

OPENCT is the maximum of files in the group open at the same time, BUFCT is the number of buffers to be reserved for the group, SVN is the prefix distinguishing a Reserve Group control word, and SIX is the prefix distinguishing an Internal Group control word.

## **Opening Reserve Files**

Each Reserve file must be opened before it can be used. It is opened by the calling sequence

_1	-* FOR RE	AΝ	RKS		_
П	LOCATION		OPERATION	ADDRESS, TAG, DECREMENT/COUNT	이
ľ	1		6 14 1		1
H	12 6	۲	0 (4)	710	= }
L	1		$T_1S_1X_1$	O.P.E.N., 4	.(
			P	<b>F.J.D.E.</b>	.)
	1				J

where

FILE is the symbolic location of the first word in the File Control Block for the file, and

the prefix, P, specifies a rewind option:

P = PZE, rewind before opening

= MZE, no rewind

= MON, no rewind, no label action

A Reserve file may be opened, closed, and reopened as many times as desired, whereas an Immediate file may be opened and closed only once.

## Section 4—File Processing

The READ and WRITE routines are used to process information files. File processing by these two routines is controlled by IOCS commands which are part of their calling sequences. The format of an IOCS command is identical to that of a machine I/O command, and its interpretation is quite similar. IOCS provides two methods of processing data. One method is to transmit between the buffers and working storage; the other is to process the data within the buffers by determining the position of the desired data. In some applications, the logical simplicity of transmitting is offset by the transmission time required; in others, the speed advantage of locating may be counterbalanced by its greater complexity or by the lack of available buffer space. Both methods may be useful and desirable in the same data processing application.

It should be noted that the meanings of the terms *read* and *write*, as used with rocs commands, differ from the conventional meanings and are used only to imply analogous functions. When used in this way in this manual, the terms are enclosed in quotation marks.

#### **IOCS** Commands

The format of an iocs command is:

S 2	3 17	18	19	20 21		35
ОР	m	*	N		A	

This is represented symbolically as:

* FOR REM	ARKS	
LOCATION	OPERATION. ADDRESS, TAG, DECREMENT/COUNT C	5)
	7 8 14 15 16	1
		= [
<del></del>	I.O.X.Y(N\*)   Am.	-(
لنسيا		}

In this representation:

- 1. X indicates the type of control, which may be one of the following:
  - C—Count Control. The count supplied with these commands specifies the number of words to be processed.
  - R-Buffer (Physical Block) Control. These commands process all (or the remainder) of a buffer.
  - S—Special Count Control. These commands process under count control except that special action (described in each case for READ and WRITE routines) is taken at the end of a buffer.
- 2. Y indicates the function to be performed at the completion of the command. Y may be one of the following:
  - P-Proceed to the next command.
  - T-Terminate the command list.
  - D—Terminate the command list and truncate any attached buffer. When an input buffer is truncated, the remainder of the block is discarded. When an output buffer is truncated, the size of the block written is equal

to the number of data words in the buffer, which need not be the same size as the buffer.

- 3. N, if used, specifies that the command is "non-transmitting" (see page 13).
- 4. \* is used, if appropriate, to specify that the address A is indirect. The effective address computation is performed as though all the index registers contained zero.
- 5. A is the address of the first word processed. If the command specifies transmission (not an ioxyn type), the address A is supplied by the programmer. If the command is non-transmitting, the system supplies the location of the next available word in the buffer. Normally, the location replaces the address of the command. However, if the non-transmitting command is indirectly addressed, the location of the next available word replaces the address of the cell specified in the address of the iocs command.
- 6. m is the number of words processed. This is normally supplied by the programmer; however, if buffer control is used for "reading," the count is supplied by the system.

IOCS TRANSMITTING COMMANDS

The transmitting commands, as the name implies, cause movement of information between buffers and working storage. The execution of each iocs command, of course, depends upon the previous commands executed. In general, each file is treated as a continuous string of words. For example, if thirteen words were "read" by one iocs command, the next command given for the file will start to "read" the 14th word, etc.

Count Control
IOCP
IOCT
IOCD

_	Г	-* FOR REM	MΑ	RKS			
	ļ,	LOCATION		OPERATION		ADDRESS, TAG, DECREMENT/COUNT	co
L	Ц	2 6	7	8 14	15	16	)
				I,O,C,Y,		A.,, m,	
	-				1		

m words are "read" or "written" (transmitted from one or more buffer(s) connected to the file into consecutive locations starting with A, or from consecutive locations beginning with A into the buffer(s)). Transition from buffer to buffer is automatic. IOCD, in addition, truncates the processed buffer when transmission is complete. (See the Glossary for a definition of "truncate.")

Buffer Control
IORP
IORT

ļ	LOCATION		OPERATION		ADDRESS, TAG, DECREMENT/COUNT	CC
-	2 6	7	8 14	15	16	
	<u> </u>		IOR,Y		A,,,,**	
						1

READ—Words are "read" (transmitted from the buffer in use by the file to consecutive locations starting with A) until the end of the buffer is reached. The count of the number of words "read" is filled into the long count field (m).

_	.[=	-* FOR REMARKS					
١	þ	LOCATION		OPERATION		ADDRESS, TAG, DECREMENT/COUNT	co
1	.!		_				)
ŀ	H	- 0	ŕ	8 14	10	Te	<del>-</del> (
ŀ	4			I,O,R,Y,	_	Am.	(
L	j					; <del>'                                    </del>	الب

WRITE—Words are "written" (transmitted from consecutive locations starting with A) until the buffer is full or the count (m) is reduced to zero. The completed block is then written.

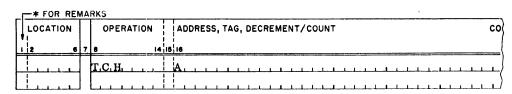
Special Count Control
IOSP

_	c	-* FOR REI	AΝ	RKS				
	ŀ	LOCATION		OPERATION			ADDRESS, TAG, DECREMENT/COUNT	co
	۱	<b>!</b>						
-	_	2 6	7	8	14	15	16	
- 1				IOSY			A m	. }
		<u> </u>						-)
L	_!		l					IJ

READ—m words are "read" into consecutive locations, starting at A, unless the end of the buffer is reached. Reading stops when the end of the buffer is reached.

WRITE—If the block being prepared is capable of holding m additional words, transmission occurs to the current buffer. If the block cannot hold m additional words, the block is written and the m words are placed in the next buffer.

Transfer and Continue Command List TCH



The command list continues at location A. The exit from an IOCs routine is to the location after the first TCH command in the list.

Examples

1. To obtain the next logical record from an input file, where the logical record is defined as 26 words, one would give the following I/O command in a READ calling sequence:

The logical record would then be "read" into locations RECORD through RECORD+25.

2. To "write" a 12-word logical record from locations ALPHA through ALPHA+11, the following command may be used:

3. Logical records may be blocked, one record per physical block, their lengths being variable. In this case, the command

would "read" all of the next physical block into locations starting at RECORD; the length of the block would be placed into the decrement of the command.

4. As a more complex example, such sequences as

can be used to process self-loading variable length records whose first word itself is an iocs command to load that record.

#### Non-transmitting Commands

Data can be processed within the buffers by not transmitting the data at all, but by *locating* it instead. Words are "located" in the buffers by using IOXYN commands.

In most cases, the IOXYN commands operate exactly like the corresponding IOXY, commands except that no words are transmitted. The location of the processed words is instead filled into the address of that IOCS command. As in example 1 above, one could use

and iocs would replace \*\* with the location of that record in an input buffer.

Clearly, locating words within the buffers adds restrictions to file design and to the IOCS command sequences. For example, one cannot locate a logical record that overlaps a physical block, since the record will not occupy consecutive memory cells. The following statements define the allowable locating functions and IOCS actions:

- 1. In "reading," words located by an IOXYN command are considered used when the file is next referenced by any IOCS routine. Hence, the buffer in which the words were located is retained until then.
- 2. In "writing," the space located for output words is considered filled when the file is next referred to by any locs routine. Hence, it is not written until then.
- 3. No single command is allowed to locate words in more than one buffer.
- 4. A sequence of commands may locate words in more than one buffer if sufficient buffers have been allocated to that file by defining a Reserve Group which consists of that file only.

The non-transmitting commands can be used to perform two functions:

- Locating, which consists of (1) determining the location in a read buffer of
  the first word of a group (also in some instances, the number of words in
  a group); (2) finding buffer space in which to place information to be
  written. Locating provides the means for processing information in the
  buffers.
- 2. Skipping, which is either (1) the omission of information "read" as input, or (2) informing 10cs of the actual length of an output record already placed in the buffer.

IOCS determines whether a non-transmitting command is intended to locate or to skip information, in accordance with the end of buffer (EOB) switch specified in the calling sequence for READ and WRITE. If EOB = 0, the command is interpreted as a skip, and if EOB  $\neq$  0, it is interpreted as an attempt to locate information. The EOB switch is interrogated when the end of a buffer is reached.

Count Control
IOCPN
IOCTN
IOCDN

-	LOCATION		OPERATION	•		ADDRESS, TAG, DECREMENT/COUNT	со
-	2 6	7	8	14	15	16	
			I.O.C.Y.N.			*,*,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,	
ĺ	1						

READ—The next m words of the file are located or skipped, and the location of the first of these replaces the \*\* in the command. IOCDN, in addition, sets an

indicator that will cause the buffer to be truncated the next time the file is referred to.

WRITE—The \*\* is replaced by the location of the next available position in an output buffer in use by the file. It will be assumed that m words have been placed into the locations specified, before the next reference to that file. IOCDN, in addition, will cause the buffer to be written upon the next reference to that file.

Buffer Control
IORPN
IORTN

_	* FOR REM	Α	RKS	
	LOCATION		OPERATION ADDRESS, TAG, DECREMENT/COUNT CO	(ر
Ŀ	2 6	7	8 14 15 16	-
			IO,R,Y,N,	1
L				

READ—The location of the next data word from the file replaces the address of the command, and the count (m) of the remaining available words is inserted into the decrement. The next reference to the file will cause the m located words to be bypassed.

ſ	1	-* FOR REI	AA	RKS OPERATION	_		ADDRESS, TAG, DECREMENT/COUNT	со
L	ı	2 6	7	8	14	15	16	i
			Г	IORYN,			* * , , , , , , , , , , , , , , , , , ,	1 1 1
L								

WRITE—The location of the next available position in an output buffer replaces \*\* in the IORYN. IOCS adjusts the available word locator m words, unless the end of the buffer is reached first. The next reference to the file will cause the contents of the buffer to be written. The user may examine the History Record (see page 17) to determine if less than m words can be placed into the buffer.

Special Count Control IOSPN IOSTN

_	* FOR REM	AΑ	RKS		
1	LOCATION		OPERATION	ADDRESS, TAG, DECREMENT/COUNT	co
	1 2 6	7	8 14	15,16	}
L	<u> </u>		IOSYN,	! !* *	

READ—Words in the buffer are skipped until either m words have been located, or the end of the buffer is reached. The address \*\* is replaced by the location of the first of the words located.

The actual number of words skipped over is reflected in the History Record (see page 17).

WRITE—If the buffer is capable of holding m additional words, the \*\* is replaced by the location of the next available word. Otherwise, the buffer is truncated and the location of the first available word of the next buffer replaces the \*\*. These commands never adjust the available word locator. Hence, an iocyn or iornyn must be executed to advance the locator by the actual number of words placed in the buffer when this number is known.

**Examples** 

1. Suppose an entire block from a given input file is to be located. Then the command

IORTN \*\*,, \*\*

should be used. When an exit is taken from the READ routine the command would have been changed to

#### IORTN RECORD, n

where RECORD is the location of the next block of the file and n is the length of the block.

2. Suppose the next word of a file is to be placed into the accumulator. Then the following command could be given:

where CLAI is the symbolic location of some particular CLA instruction. The CLA would be modified as follows:

before READ: CLAI CLA \*\*
after READ: CLAI CLA LOC

where LOC is the location of the desired word.

3. Suppose it is desired to locate space in an output buffer in which to create a variable length output record. The command

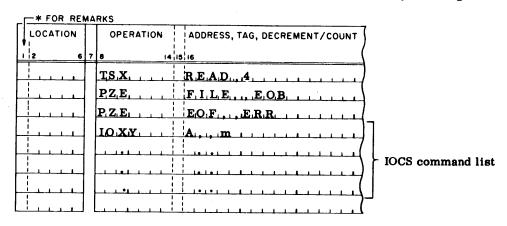
will locate an area of MAX number of words, and the location of that area will replace \*\*. The variable length record would now be created by the user's own program. Inasmuch as the command was an IOSYN (the only command which does not advance the available word locator), he must subsequently inform IOCS of the actual number of words placed in the located area by means of a skipping command such as

where ACT is the actual number of words placed in the output buffer, that is, the length of the variable length record.

#### **IOCS** Routines

**READ** 

Words are transmitted from, or located within, the buffers by the sequence



#### where

FILE is the file designation, EOB is the end of buffer switch, EOF is the end of file exit, and ERR is the error exit.

The command list is terminated by the first IOXT, IOCD, IOXTN OF IOCDN command encountered.

- 1. Eof is the location to which transfer is made when an end of file condition occurs. For a labeled file, the condition is recognized from the trailer label. For an unlabeled file, any Eof mark is recognized as end of file. For any file, recognition of the Eof mark suspends buffering, so that there is no information for the file in any buffer when the Eof exit is taken. Buffering will be restarted when the next READ (if any) is given for the file.
- 2. ERR is the location to which transfer is made when any of three types of error conditions occur: (a) a redundancy which cannot be corrected; (b) check sum error (binary file); and (c) sequence error. The condition is recognized at the first reference to a buffer in which it occurs. The error encountered may be ignored by continuing to "read" the file.
- 3. The end of buffer switch (EOB) is interrogated each time the end of a buffer is reached, regardless of whether a transmitting or non-transmitting command is being executed.
  - a. If EOB = 0, truncation of the buffer and automatic transition to the next one occur; command execution continues without interruption.
  - b. If EOB  $\neq$  0, all information located will be retained until the next reference to the file by any iocs routine. Further, since information located by a *single* iocs command must be in sequential cells, the execution of a count command interrupted by the end of buffer condition is discontinued, and EOB itself is used as the exit of transfer address. Otherwise, transition to a new buffer is automatic, and the interpretation of the command sequence continues.

Non-transmitting commands may be freely intermixed with transmitting commands in any command sequence. However, since the EOB switch is set on or off by each entry into the READ or WRITE routines, skipping and locating cannot be done by the same sequence.

WRITE

Words are transmitted to, or space for words is located within, the buffers by the sequence:

		-* FOR REM	14	RKS				
		LOCATION		OPERATION	1	ADDRESS, TAG, DECREMENT/COUNT		
	ı	2 6	7	8 14	115	16		
				T.S.X.		WRITE, 4	١	
ļ				P,Z,E, , ,		F,I L,E,,,, E,O,B,		
		<u> </u>		10X.Y	<u> </u>	<u>A.,,,,,,,,,,,,,,,,,,,,,,,</u>	1	
		<u> </u>			<u>!</u>	101011111111111111111111111111111111111		
		<u> </u>			<u>i</u>		}	IOCS command list
	_	<u> </u>	Ì	<u></u>	<u>i</u>	<u> </u>	)	
1		i ! ! ! ! ! !			<u>i</u>	<u> </u>		

The command list must be terminated by an IOXT, IOCD, IOXTN or IOCDN command. The EOB switch functions exactly as described under READ.

#### HISTORY RECORDS

When the normal exit from the READ or WRITE routine occurs, a record of action, similar to the record produced by a Store Channel instruction, is supplied, as explained below. This "history record" is provided to supply the user with information concerning the last iocs command that was executed. The count of the remaining words in the buffer can be used to create future iocs commands or to make other logical decisions. The investigation of the last word "read" or "written" may be necessary to determine what the command actually did, as in the case of iost in reading, where the count specified may not have been satisfied because the end of the buffer was encountered.

Similarly, IORY, in writing, may not write the specified number of words if it encounters the end of buffer condition.

At each exit from the READ or WRITE routines the AC will contain the following:

Bits	Contents
3–17	Number of usable words remaining in the buffer which contained the last word "read," or the number of unused words in the buffer containing the last word "written."
21–35	1 plus the location of the last word transmitted.

If an end of buffer, an end of file, a sequence error, a check sum error, or a redundancy which cannot be corrected occurs while reading, or an end of buffer occurs while writing, the following information is provided:

Register	Bits	Contents
AC	3–17	The 2's complement of the quantity (1 plus the location of the command being executed when the condition was encountered).
AC	21–35	1 plus the location of the last word "read" or "written."
MQ .	3–17	The 2's complement of the location of the TSX to the READ or WRITE routine.
MQ	21–35	Location of the normal return from the 10cs routine.

In addition, when a redundancy, check sum, or sequence error occurs during reading, the prefix of the MQ will contain:

- 6—if check sum and redundancy errors occurred.
- 5-if sequence and redundancy errors occurred.
- 4-if a redundancy occurred which could not be corrected.
- 2-if a check sum error occurred (see page 33).
- 1-if a sequence error occurred.

The order in which these errors are detected is: check sum, block sequence, and redundancy. If all three checks are being made on a file, the occurrence of only a redundancy error is probably a *false* redundancy error, as the check sum is correct; however, the user must realize that check sums are not a foolproof check.

When an EOB exit occurs during either reading or writing, the prefix of the MQ will be:

- 4-if all available buffers are in use.
- 2-if the end of buffer condition was encountered during execution of an IOCYN.

Transfer of Data COPY

Information which was located by the last I/O command sequence on the input file FILE1, may be transferred to the output file FILE2 without actual word transmission by:

_	* FOR REM	MΑ	RKS		
	LOCATION		OPERATION	ADDRESS, TAG, DECREMENT/COUNT	cc)
Ľ	2 6	7	8 14	15 16	
			T. S.X.	C,O,P,Y,,,4,	
L			P.Z.E.	FILE1., FILE2	\
	1		<u> </u>	<u> </u>	لب

The rules for information transfer with copy are:

- 1. All words in the same buffer with, and preceding, the first word located are included in the output.
- 2. All words in the same buffer with, and subsequent to, the last word located are not included in the output; furthermore, they will behave as if skipped when filel reading is resumed.
- 3. If more than two buffers are involved, an intermediate buffer is included in the output provided at least one word within it has been located by a nontransmitting command.

Transfer of information between an internal file and some other file, is accomplished by:

* FOR REMA	ARKS		
LOCATION	OPERATION	ADDRESS, TAG, DECREMENT/COUNT	ď
112 - 67	8 14	15,16	1
	T.S.X.	S.T.A.S.H., 4	7
	PZE	F.I.L.E.1., F.I.L.E.2	7
	P.Z.E.	N.T.S.	7
	L	; <del>-!- </del>	7

where NTS is an exception exit (similar to EOB). Discussion of the STASH routine is given in Section 8.

Other System Routines CLOSE

STASH

When activity on a file is to be terminated, the sequence

_	- * FOR REM	<u> 1 A</u>	RKS			
1.	LOCATION		OPERATION		ADDRESS, TAG, DECREMENT/COUNT	co
	2 6	7	8 14	15	16	\
			T, S, X,		$C_1L_1O_1S_1E_1$ , $\frac{4}{3}$ , $\frac{4}$ , $\frac{4}{3}$ , $\frac{4}{3}$ , $\frac{4}{3}$ , $\frac{4}{3}$ , $\frac{4}{3}$ , $\frac{4}$	{}
L	<u> </u>		P		F.I.L.E.	{
L						لب

will insure that the file does not unnecessarily withhold buffers from use. P determines the type of closing as follows:

P = PZE Close with rewind-unload (Note: a file which is rewound and unloaded may not again be referenced.)

= PTW Close with rewind = MZE Close no rewind

= MON Close no rewind, no EOF mark or trailers written.

With any type of close, a delay occurs until all I/O activity on the file ceases, forcing all buffers in use by the file being closed to be released to the pool. The only routine which can produce any further effect on any closed file is an open. If the file is an Immediate file, the buffer availability count, which keeps track of the number of available buffers in the pool, is increased. If the file is a Reserve or Internal file, the count is not increased until its associated Group control word appears in a close list. For a regular output file, a close other than the MON type, causes writing of (1) an EOF mark, a trailer label, and another EOF mark, if the file is labeled, or (2) an EOF mark only, if the file is unlabeled.

More than one file can be closed at a time by means of the calling sequence:

_[.	* FOR RE	MΑ	RKS																			
$\prod$	LOCATION	l	OPERATION	11	ADDRESS,	TAG,	DEC	REM	ENT.	/co	UNT										C	o
ľ	12 6	7	8 14	15	16																	_/
			T,S,X, , ,		C,LOS,	E., .	4. ,			1 1.			 									)
L			P	Ш	LI,S,T		F,I,	$L_{L}\mathbf{E}_{L}$	CiT		1.1	1	 		_1			1		1		7
Ŀ	<u> </u>			! ;	 		1 1	1 1	1				 ,	L		1	. 1 .	1	1. 1	1	. 1	. l

LIST and FILECT have the same meaning here as in the calling sequences of the ATTACH routine, except that this list may be only a part of the list that was used to attach the files. P specifies the rewind options as given above. The same rewind option is used for all files in the list to be closed. If a Reserve Group control word occurs in the list, the entire group which it controls must also appear. In this case, the buffer availability count is increased by the number of buffers in the group.

Non-data Functions WEF

An end-of-file mark can be written by using the sequence:

_ + FOR RE	MΔ	ARKS	
LOCATION		OPERATION ADDRESS, TAG, DECREMENT/COUNT	cc
1 2 6	, ,	8 14,16,16	
-		T.S.X. W.E.F., 4	
<u> </u>		P.Z.E. F.L.L.E., E.O.T.	
	J		٠, ١

For an output file, any buffer connected is truncated and the WEF executed. If end of tape is encountered while writing the tape mark, the end of tape exit (EOT) is taken. The operation is ignored for an input file. WEF will be ignored on a labeled file since it would result in an incorrect end of file condition for the

output file. Instead, CLOSE should be used, as that routine will automatically assure proper arrangement of the file.

REW

Similarly, a rewind can be caused by:

* FOR REM	IARKS	_
LOCATION	OPERATION ADDRESS, TAG, DECREMENT/COUNT	col
111		1
1 2 6	7 8 14 18 16	1
	T, S,X, , , , , , , , , , , , , , , , , ,	
1	1 1	
<del>                                     </del>	P,Z,E, , ,   F,I L,E, , , , , , , , , , , , , , , , , ,	
11		J

All buffers in use by the file are released to the pool and the rewind is issued. An attempt to rewind a labeled file will be ignored.

BSR

Backspace of a *physical* block can be accomplished by:

LOCATION		OPERATION		ADDRESS, TAG, DECREMENT/COUNT	cq
1 2 6	7	• 14	15	16	
		т, s,х,	!	B.S.R., 4	٠,
	1	PZ.E.	!	F.I.L.E., BO.F.	
	1		!		

The file is first unbuffered, and an actual machine BSR instruction is then given. However, if the BSR would pass over an EOF mark, it is suppressed and the beginning of the file exit (BOF) is taken.

Backspace of a specified number of physical files can be accomplished by:

* FOR RE	MA	RKS	
LOCATION		OPERATION	ADDRESS, TAG, DECREMENT/COUNT CO
1*:	ı	]	
112	6 7	8 14	15 16
		T.S.X.	BS.F., 4.
		PZE	F.L.L.ECOUNT

where COUNT is the number of files to be backspaced. The file is first unbuffered, and "COUNT" machine BSF instructions are given, followed by a forward space over the last EOF mark encountered. However, the forward space is suppressed if the tape reaches the rewind position. A COUNT of zero can be used to simply unbuffer the file.

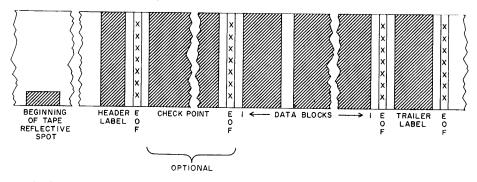
## Section 5 — Labels and Labeled File Procedures

IOCS provides a standard tape labeling system as a specifiable option. The IOCS labeling procedure provides, as do most labeling procedures, a method for magnetic tape identification and automatic multi-reel file handling. Each reel of tape has one "header" record (see page 24 concerning additional header records) at the beginning of each reel which serves to identify the reel as belonging to a particular file. Each reel also has a "trailer" record which indicates whether this is the last reel of a file or not.

If labeled files are specified, IOCS will automatically provide reel switching, file identification messages as the file is prepared, and protection for files which are to be saved.

## Format of Labeled Tapes

The format of a labeled tape is as follows:



- 1. Labeling is not available for files processed on any on-line card equipment.
- 2. Header labels are always written in the BCD mode and may be in either high or low density.
- 3. The EOF mark following the header label is in the same density as the label.
- 4. The optional checkpoint record and the EOF following it are in the same density as the file.
- 5. The file itself may be either high or low density.
- 6. The EOF mark, the trailer label, and its EOF mark are always written in the BCD mode in the same density as the file itself.

#### Standard Label Formats

BLANK REEL

iocs will automatically process 14-word labels of the following formats:

A temporary header label can be placed on an otherwise blank reel by a card-totape operation. It should have the following format:

	Word(s)	Card Column(s)	Content	Description
	1	1–6	1BLANK	Label identifier
	2	7–12	bXXXXX	Tape serial number
	3–14	13–80	Arbitrary	Not used
where:				

1BLANK serves to identify the reel as a blank reel available for use.

TAPE SERIAL NUMBER identifies the reel by number. It is assumed that each installation has a standard tape serial numbering system.

HEADER LABEL	Word(s)	BCD Position(s)	Content	Description
	1	1- 6	1HDRbb	Label identifier
	2	7–12	bXXXXX	Tape serial number
	3	13–18	bXXXXX	File serial number
	4	19–24	bXXXXb	Reel sequence number
	. 5	25–30	XXbYYY	Creation date XX — Year YYY — Day of year (001-365)
	6	31–36	bbbXXX	Retention days
	7	37–38	bX .	$ \begin{array}{c} \textbf{File density} \\ \textbf{0-Low} \\ \textbf{1-High} \end{array} $
,	7	39	X	File mode 0 — BCD 1 — Binary
	7	40	X	Block check sum indicator (Binary files only) 0 — No check sums 1 — Check sums present
	7	41	X	Block sequence indicator (Binary files only) 0—No sequence word appears 1—Sequence word is present
	7	42	X	Checkpoint record indicator  0 — No checkpoint record  1 — Checkpoint record follows label
	8–10	4360	$X.\dots X$	BCD file name
	11–12	61–72	Arbitrary	Not used, but reserved for 709/90 Sort compatibility
	13–14	73–84	Arbitrary	Not used

#### where

TAPE SERIAL NUMBER identifies the serial number of the physical reel. It is recommended that the number which externally identifies the reel be used as the tape serial number for the label.

FILE SERIAL NUMBER is the tape serial number of the first reel of the file. This number is taken from the label of the first reel on which the output file is prepared.

REEL SEQUENCE NUMBER is the number of the reel within a given file (i.e., the first reel of a file is reel 0001, the second is reel 0002, etc.).

CREATION DATE and RETENTION DAYS are the date the file was written and the number of days it is to be retained. If an attempt is made to use

this reel before (Creation Date + Retention Days) has been reached, 1000 signals the label error by the message "RETENTION NOT EXPIRED."

TRAILER	LABET.

Word(s)	BCD Position(s)	Content	Description
1	1- 6	1EORbb or	End of reel trailer
		1EOFbb	End of file trailer
2	7–12	bXXXXX	Number of blocks in file
3	13–18	bXXXXX	Unit control word
4-14	19–84	Arbitrary	Not used

#### where:

1EOR will cause tape switching to occur automatically during reading. No EOF indication is given the user when an EOR trailer is encountered.

1EOF will cause an EOF exit from the READ routine If the file is again referenced, another EOF indication is given.

BLOCK COUNT is the number of data blocks on this reel. This is checked for every input reel. If it differs from the number of blocks actually read in, a sequence error is given. In this case it should be noted that a sequence error for the reel can occur even though each block is not being checked for sequence.

UNIT CONTROL WORD is the location in storage of the first word of the Unit Control Block which specifies the tape unit the reel was prepared on, and the logical number it was assigned.

Note: During writing, an EOR label is prepared by IOCs when the end of a reel is encountered. An EOF label is prepared when the file is closed.

## Labeling and Label Checking

INPUT FILE HEADER LABEL

When the file is opened, and at the beginning of each new reel, if it is a multi-reel file, the label is checked to ensure that the correct reel is being processed. The following conditions must be met for an input label to be valid:

- 1. The file serial number given on the FILE card must agree with the file serial number in the label.
- 2. The reel sequence number in the label must agree with the reel sequence number given on the FILE card, unless, of course, reel switching has occurred, in which case IOCs will have advanced the reel sequence number stored in the File Control Block.
- 3. The file name given on the FILE card must agree with the file name in the label. If no file name is given on the FILE card, this test is bypassed.

If any of the above checks fails, and the label search option has not been specified, the message LABEL ERROR, INPUT REEL INVALID, and the label in error are printed. A halt will then occur. By sense switch control, the operator may accept the reel as valid, in which case he may proceed; otherwise he must mount the correct reel to be processed.

If the label search option has been specified, the message (file name) SEARCH SKIP LIST FOLLOWS is printed, and searching occurs until the correct label is found or the operator intervenes.

Additional Header Records on Input Files

The system will automatically bypass all records between the header label and the next for mark. The system provides no automatic way to create these records and this feature is provided for compatibility with existing labeling schemes. If these records are to be processed, the file should be specified as unlabeled; the user can then process the label records as he desires.

INPUT FILE TRAILER LABELS

Each labeled file is assumed to be a multi-reel file. The occurrence of an eor trailer causes tape switching to the next reel, which is checked before processing is continued. The occurrence of an eof trailer or, in fact, any record other than an eof trailer where a trailer label is expected, will cause an eof exit to be taken from the READ routine. The Block Count which appears in every trailer written by the system is checked and a sequence error exit is taken if it does not agree with the number of blocks "read."

OUTPUT FILE HEADER LABELS

Every reel upon which a labeled output file is to be prepared must have a label written on it. This label may be a 1BLANK type or a 1HDR type on which (Creation Date + Retention Days) has been reached. If these conditions are not found, either or both of the following messages will be printed:

LABEL ERROR, OUTPUT REEL INVALID LABEL ERROR, RETENTION NOT EXPIRED

The label is then printed and a stop occurs. Either condition may be ignored by the operator, in which case a dummy label is generated, with the file serial number \*\*\*\*\*, and 10cs continues.

OUTPUT FILE TRAILER LABELS

If, during the course of writing an output reel, the for reflective spot is sensed, an for mark, an for trailer, and another for mark will be written. The reel will then be rewound (and unloaded if a 7090 is used) and a message will be printed to the operator to remove that reel. If the file is assigned two different units, reel switching occurs and processing continues. Otherwise, a delay will occur until the present reel is removed, and a new labeled blank is mounted.

#### Non-Standard Labels

Non-standard labels may be either a mere arrangement of the standard format or an entirely different labeling procedure. If non-standard labeling procedures are desired, the rocs labeling routines must be entirely replaced. However, if only changes in label formats are desired, the rocs label handling routines will still apply.

If the non-standard labels to be processed adhere to the general IOCS labeling scheme, non-standard label images may be easily handled by IOCS, using different routines to check and prepare label images. For this purpose, IOCS allows a non-standard label image package to be specified for each file. The location of the first cell of this package can be specified in the decrement of each entry in the list of files used for the ATTACH routine. If the decrement is zero, standard images are assumed; if it is non-zero, the following procedures will be executed:

- 1. All labels, both headers and trailers, will be read and written by the system.
- 2. All EOF marks associated with labels will also be read and written by the system.
- 3. The standard label area at symbolic locations 10cs + 43 (see page 39) will be used; its length is 14 words.
- 4. All the system actions and messages will apply.

- 5. At each TSX, index register 2 will contain the 2's complement of the location of the first word of the File Control Block for the file being processed.
- 6. The contents of all index registers used must be saved and restored.

Assuming that the non-standard label package begins at symbolic location MYLBLS, then the following actions will occur as indicated:

1. TSX MYLBLS, 1

To check header label input file

The routine must return by means of: TRA 1, 1 if label is correct

TRA 2, 1 if the input label is invalid

Bit 35 of the 7th word of the label area must be 1 upon either return if a checkpoint record is to be skipped.

2. TSX MYLBLS + 1, 1 To check trailer label, input file

Return must always be: TRA 1, 1 Upon return, the sign of the MQ must be:

+ if EOR; tape switching will occur

- if EOF; an EOF exit will be taken

3. TSX MYLBLS + 2, 1 To check label on tapes to be used for output

Return must be: TRA 1, 1 if the reel may be used

TRA 2, 1 if (creation date + retention days)

has not been reached

TRA 3, 1 if it is not possible to use the reel

4. TSX MYLBLS +3, 1 To prepare output trailer label

Return must always be: TRA 1, 1

Upon return, bits P, 1-35 of the AC must contain:

1EORbb if an EOR trailer is to be prepared 1EOFbb if an EOF trailer is to be prepared.

5. TSX MYLBLS +4, 1 To prepare output header label

Return is: TRA 1, 1

#### Section 6-Unlabeled File Procedures

Although the handling of labeled files is entirely automatic, IOCS is equally capable of processing unlabeled files. The difference is that multi-reel file handling is only semi-automatic for unlabeled files, since the occurrence of the EOF mark can have several meanings.

### Single-Reel Unlabeled Files

An unlabeled file which is contained on one reel of tape may have any number of EOF marks. The detection, during reading, of each EOF mark temporarily suspends buffering on that reel. When an EOF is detected by the READ routine, the EOF exit is given. If the file is again referred to by a READ calling sequence, buffering will continue until the next EOF mark is encountered. It is the programmer's responsibility, in this case, to determine which EOF mark signifies the actual end of the file. No reel switching is possible for input files of this type. If EOT is encountered while writing a supposedly single reel unlabeled file, a reel switch will occur and processing will continue. In some instances, such as peripheral output files, no harm occurs from this action. However, if the file is to be processed by IOCS at a later time, the file is not describable. (It is a multiple-reel file with more than one EOF mark on a reel, a situation which is not allowable.)

#### **Multi-Reel Unlabeled Files**

A multi-reel unlabeled file may have only one EOF mark—that which signifies the end of the reel. This EOF mark is written automatically when the EOT reflective spot is sensed while writing. Reel switching occurs after the EOF is written. When an EOF mark is detected while reading a multi-reel unlabeled file, all buffering is suspended until the EOF record is reached. An EOF exit is given once per reel for each EOF encountered and if the file is again referenced by a READ operation, reel switching will occur and buffering will be resumed. For this type of file, the user must have a recognizable data record to indicate that the end of the file has been reached.

#### **Multi-File Reels**

Several files which appear on the same reel may be processed automatically by closing each processed file without rewinding and then opening the subsequent file. Iocs does not automatically rewind any reel except at tape switching. Hence, a single reel may contain several complete files and the first portion of a multireel file. Labeled and unlabeled files may be mixed upon the reel, provided some precautions are observed in the usage of Iocs routines. For example, the BSF routine with a count greater than 1, may not result in positioning at the beginning of data of a given file because of the occurrence of EOF marks for labels, trailers, and checkpoint records. It is also unlikely that OPEN with label search would find the correct file.

## Section 7-Use of the Reserve Group Option

An unusual feature of rocs is that the programmer does not normally assign a fixed set of buffers for the exclusive use of any file. This can be done by attaching only a single file to a pool, but at the almost certain sacrifice of efficiency in time, space, or both. The attachment of more than one file to a pool, however, does create a problem: the system must ensure that any file using a pool will be able to obtain at least one buffer when needed, since otherwise reading or writing could not be accomplished. In rocs, control is exercised by means of the Buffer Availability Count (see page 75), and for most data processing applications this is sufficient.

Sometimes, however, situations occur for which more control is either necessary or desirable. The Reserve Group option is the programmer's principal tool for these "unusual" cases; and an understanding of its use is necessary to obtain maximum utility of iocs. The paragraphs which follow discuss three situations and illustrate the use of the Reserve Group in handling them.

## Deferred Opening of a File

If a file is to be used late in a program, or perhaps not at all, depending upon the flow, then placing it in a Reserve Group will generally benefit the program, since buffer space will be free until needed by the file. For example, if file1 and file2 are the respective locations of the File Control Blocks for two files which will never be used at the same time, attaching them to a pool as the Reserve Group:

_	* FOR REM	AN	RKS		
	LOCATION OPERATION		OPERATION	ADDRESS, TAG, DECREMENT/COUNT	
<u> </u>	2 6	7	8 14		1
L			S.V.N.	1,,,,1	マ
L			PZE	F.I.L.E.1	7
			P.Z.E	F,I,L,E,2,	
L	<u> </u>	i		 	7

will withhold only one buffer from the general pool availability, whereas attaching them as Immediate files would withhold two.

## Using a File for Both Input and Output

In some instances, a file is used both as input and as output. It is not unusual, for example, to create an intermediate tape as output from one phase of a problem, and then to use this tape as input to some later phase. A simple way to handle this is to refer to the file using separate File Control Blocks for the input and output phases. That is, identical blocks should appear at locations in and output phases. That is described as an input file and out as an output file. The file can then be attached as a Reserve Group, as follows:

FOR REM	ARKS	
LOCATION	OPERATION	ADDRESS, TAG, DECREMENT/COUNT CO
1 2 6	7 8 14	15 16
	S,V,N,	1,,,,1,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,
	P.Z.E.	I , N,
	P,Z,E	OUT

The our file would be opened and used during any output phase, while the IN would be closed; the reverse would apply during an input phase.

## Using More Than One Buffer for a File

If a file is being used in such a way as to withhold more than one buffer at a time from a pool, then it must be treated as a Reserve Group (by itself) and the buffer requirements expressed by a sufficiently high count.

Suppose, for example, that a file, whose File Control Block is located at file, has been produced by a card-to-tape operation. Suppose, further, that five cards of this file are to be processed at a time, and that the information from these five cards must be retained during the processing (sorting is such an example). A convenient way of reading the file is by:

* FOR REMA	ARKS		
LOCATION	OPERATION	ADDRESS, TAG, DECREMENT/COUNT	co
1 2 6 7	8 (4	15116	_]
	T, S, X,	R.E. A.D., 4	
	P.Z.E.	F, I, L, E, , , , E, O, B	J
	P,ZE	EOF, , , ERR	$\perp$
	I O R PN.*	R.1.,,**	لــا
	IORPN*	R2,,,**	_/
	IORPN*	R3,,**	$\Box$
	IORPN,*	R4,,,**	
	IORPN*	R, 5, , , * *	7.
			$\overline{\cdot}$

In this READ usage, since EOB  $\neq$  0, the non-transmitting IOCs commands locate the initial address of each card buffer. Since indirect addressing is used, these addresses are placed by IOCs into the addresses of the words located at R1, R2, R3, R4, and R5, respectively. If, then, these words were

_	Γ	* FOR RE	MΑ	RKS				
	Į.	LOCATION		OPERAT	ON	i	ADDRESS, TAG, DECREMENT/COUNT	col
L	`	2 6	7	8	14	5	16	-
L		R <sub>i</sub> ,		AXC.		l	*,*,,4,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,	7
L	-					1		$\overline{}$

then the Nth word in any of the buffers can be referenced by first executing the appropriate instruction at location  $R_{i,}$  and then executing:

_	_	* FOR REM	<u> 1 A</u>	RKS																	 								_
1.	ŀ	OCATION		0	PE	RAT	,IOŅ		ļΑ	DDR	ES	s, 1	Ά	3, D	ECR	EM	EN	T/0	:ou	NT								C	(
L		2 6	7	8				4 15	16																				)
	I			O.F					N	•	4,	1									 			_					3
L	1	1 1 1		L				1	L					· 				1			 	 	1		L	 اا	 		}

It is clear that this procedure will require the use of no fewer than five buffers by the file at all times. Actually, iocs would try to use at least ten, for at each entry to the READ routine, when the five buffers just processed are released to the pool, minimum delay (that is, maximum overlap) occurs if the next five have already been filled and are waiting to be located. If, however, ten are not available—either because there are not ten in the pool, or because other files are using them—the system can still operate properly as long as there are at least five available to the file. This condition can be guaranteed only by an attachment of the form:

FOR REA	MARKS		
LOCATION	OPERATION	ADDRESS, TAG, DECREMENT/COUNT	cd
1 2 6	7 8 14	15 16	
	S.V.N.	1,,,M,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,	
	P.Z.E.	F,I,LE	
		; . <del> </del>	

with M not less than 5.

Finally, it should be noted that if the programmer had erred and written:

_	-* FOR REI	AΝ	RKS			
	LOCATION		OPERATION		ADDRESS, TAG, DECREMENT/COUNT	cd
L	1 2 6	7	8 14	15	16	
			S,V,N		1,,,3,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,	)
L	<u> </u>		P,Z,E		F,I,LE	L.I.L.
L				!		<u> )</u>

then the eob exit would have been taken after execution of the third iorpn\*, and this would occur even if, at that particular time, a buffer were available in the pool for use by another file. iocs control forces this action, since otherwise some file might later be unable to obtain a buffer. In this usage, eob is to be interpreted as "end-of-available-buffers."

# Section 8—Use of the Internal Group

An Internal Group is a special type of Reserve Group. Its files are retained entirely in core storage (in buffers within the pool allocated to the group) and are never, as such, written onto, or read from, an I/O unit.

The programmer, however, processes these files in the same way as a regular file. The full power of the READ and WRITE routines is available, and their usage is the same. Because of the special nature of Internal files, some system routines have no meaning (and so do not apply), while others are interpreted in a manner consistent with this nature.

Use of the Internal file can result in increased efficiency for some applications. In others, such as storage allocation and searching problems which arise in the generation and manipulation of variable length tables, it can prove of substantial benefit.

# **Opening an Internal File**

An Internal file, like a Reserve file, is activated by the OPEN routine. Unlike a regular file, however, the open is also used to indicate the type of processing desired. In the sequence

FOR RE	MΔ	RKS		
LOCATION		OPERATION	ADDRESS, TAG, DECREMENT/COUNT	cd
1 2 6	7	14	15 16	1
1		TS X	O.P.E.N.4	
		B	F,I,LE	$\Box$
		<u> </u>		$\Box$

the interpretation of P is:

P = PZE

= PON

=MON

	ready in the file is discarded. This is analogous to opening a regular output file with rewind.
= MZE	The file is prepared for writing, and any information already within it is retained. This is analogous to opening a regular output file without rewind.

The file is prepared for reading, making any information within it available. During the course of using the READ routine, processed buffers are released to the pool exactly as for a regular file. This mode of operation can be thought of as a "destructive" read.

The file is prepared for writing, and any information al-

The file is prepared for reading, making any information within it available. However, in this case, processed buffers are never returned to the pool. They are retained by the file, and may be reprocessed on a later pass through it. This mode of operation can be thought of as a "regenerative" read.

# Closing an Internal File

The CLOSE routine is used to terminate the current type of processing of an Internal file. In the sequence:

_	* FOR REM	AΑ	RKS	
\[	LOCATION		OPERATION	ADDRESS, TAG, DECREMENT/COUNT CO
L	2 6	7	8 14	15 16
			T,S,X,	C,L,O,S,E,,,4
L			P	F,I,LE,

the interpretation of P is:

P = PZE

All buffers in use by the file are released to the pool, and the file may not again be referenced. This is a "destructive" close analogous to a rewind and unload for a regular file.

= MZE

Reorganizes the structure of the file, causing it to appear as if it had just been written. The file may be opened again (with any of the available options). This may be thought of as a "regenerative" close.

# **Processing an Internal File**

READ WRITE The only difference in the use of READ and WRITE for an internal file is in the interpretation of the meaning of the EOB exit:

- 1. The EOB exit will occur on writing when an additional buffer is needed and all the buffers allotted to the Internal Group are in use. In effect, this is an end of buffers exit.
- 2. The EOB switch on reading works exactly as for a regular file, except that the end of buffers exit *cannot* occur. For an Internal file in read status, end of buffers and EOF are synonymous.

During the course of processing, it is often desirable to transfer information between an Internal file and some other file. This is accomplished by means of the special copy routine, STASH. The sequence:

ļ	LOCATION		OPERATION		ADDRESS, TAG, DECREMENT/COUNT	cq
1	2 6	7	8 14	15	16	
			T.S.X.	- 1	S.T.A.S.H.,4	
			P.Z.E.		F.I.L.E.1F.I.L.E.2	_ 1.1
Г	1		RZE.		N.T.S.	' . السا
Γ	1					

effects the transfer by means of buffer reassignment from file1 to file2, provided both files are attached to the same buffer pool. NTS is a location to which return will be made for "Nothing To Stash" or "Noplace To Stash" conditions, under circumstances which are described below.

STASH

Four cases are permitted in the use of the STASH routine. These cases, and the IOCS actions involved, are:

1. FILE1 is an internal output; FILE2 is a regular output.

The initial buffer is deleted from the chain of file1 buffers, and appended to file2. The NTS return will occur if there is no buffer in the file1 chain which has been completely processed. This usage allows the programmer to provide a regular output file for "overflow" of the Internal Group area.

2. FILE1 is a regular input; FILE2 is an internal output.

As in the case of the COPY routine, the buffer or buffers which contain information which was located by the last read sequence on filel, are appended to the file2 buffer chain. The NTS return occurs if during the reassignment, the Internal Group is about to overflow its full quota of buffers (as indicated by BUFCT in the Internal Group control word). This usage allows the programmer, for example, to set aside some data from file1 for later processing, without the necessity of again processing the entire file.

3. FILE1 is an internal input; FILE2 is a regular output.

For this usage, the STASH routine behaves exactly like the COPY routine. It should be noted that buffers will be removed from the FILE1 chain and hence are lost to FILE1, even if the "regenerative" read mode is in use. The NTS return cannot occur.

4. FILE1 is an internal input; FILE2 is an internal output.

This functions exactly like case 2. However, buffers will be removed from the file1 chain even if the "regenerative" read mode is in use. The NTS return occurs if during the reassignment, the Internal Group to which file2 belongs is about to overflow its full quota of buffers. Note that even though file1 and file2 must be attached to the same pool, they need not belong to the same Internal Group.

# Section 9 — Block Sequence Numbers and Check Sums

Every File Control Block carries a block sequence number for the current reel of the file. On a labeled file, this sequence number is always written in a trailer label at the end of each reel and checked when it is read.

In addition, for a binary file, a block sequence word can be appended to the end of each block, regardless of whether the file is labeled or unlabeled. This word contains the sequence number of the block in its address and may also contain an 18-bit folded check sum of the block in the left half of the word. The check sum is formed by computing a logical sum of the entire data in a block, excluding its block sequence word, and then logically adding the left and right halves of this sum. This computation requires the execution of at least two instructions for each data word, and hence may be costly to the program. On the other hand, forming and checking a block sequence number is accomplished in a few programming steps for each block, and it is quite helpful in detecting certain machine malfunctions, such as a tape shift-register trouble.

Note that in order for a block sequence word to be generated and checked, each buffer requires an additional word, which must be accounted for when the buffer pool is defined. However, the generation and checking are accomplished internally and this extra word is transparent to the processing of data.

# Section 10—Checkpoint and Restart

A checkpoint is a reference point to which a program may return for restart. Restart may be necessitated for various reasons: machine failure, operator error, erroneous data, interruption of a program by an operator, etc. In rocs, the checkpoint record contains all the necessary information for restart.

A checkpoint may be initiated by one of three conditions: beginning of reel on a labeled file, execution of the checkpoint routine by a program, and operator intervention by the use of a sense switch assigned for this purpose. In the first case, a checkpoint is written either after the label on an output tape or on the checkpoint file depending on which was specified in the file card. In the latter two cases, the checkpoint is always written on the checkpoint file.

The checkpoint routine saves the contents of all index registers, and sense indicator and sense light conditions, and sense switch settings, forces cessation of all input/output activity, prints out a checkpoint identification together with a restart code, and writes a checkpoint record of two blocks followed by an EOF mark. The first block contains information about the use of files and the necessary data for tape repositioning. The second block contains the contents of all of core storage and the conditions of the registers and indicators. The checkpoint identification is a sequence number and it is increased by 1 every time a checkpoint record is written.

A checkpoint file should not be attached to any buffer pool, as it uses no buffers. However, it must be opened. iocs records which was the last checkpoint file opened and always uses that as the current checkpoint file.

A checkpoint can be written by the programmer by using the calling sequence:

_	* FOR REM	MΑ	RKS		
ŀ	LOCATION		OPERATION	ADDRESS, TAG, DECREMENT/COUNT	cd
Ŀ	1 2 6	7	8 14	15 16	)
L	<u> </u>		T,S,X,	C,K,P,T,,4,	$\overline{}$
L	<u> </u>				(

If the checkpoint routine is entered, but the checkpoint file has not been specified by the use of a file card, or been opened, immediate return is made to the program.

Restart is strictly an operator procedure and cannot be initiated by a program. The restart routine uses the information in the checkpoint record to print messages to assist an operator in mounting tapes and setting sense switches, to reposition tapes, and to restore the core storage, index registers, sense indicator and sense lights to their status when the checkpoint was taken. The execution of the program is resumed from the point in the program at which the checkpoint record, used for the restart, was written.

# Section 11 - Density and Mode Conventions

# **Density**

If a reel is to be involved in restart, rocs allows a tape density change, if any, only after the first EOF mark on the reel. The reason for this restriction is that the restart routine must know the density of every block and EOF mark to be skipped over in the process of tape positioning. If an arbitrary number of density changes is to be permitted, every change would have to be indicated by some means; this would require a rigid tape format.

Tape densities of a file and its header label are specified by a FILE card and are stored in a File Control Block. The density of a checkpoint record on a labeled file is the same as that of the main body of the file. In the case of a multi-file reel, a density change is permitted only after the header label of the first file; the remainder of the reel must be the same density because of the above restriction.

A standard density option is provided for labeled files. If this option is indicated on the file card, the density of the label will be that specified for the file. Furthermore, when an output file is prepared, the density of the label on the reel to be checked for validity is always the same, low or high according to installation choice. It should be emphasized that if the standard option is *not* used, the density of the label of the blank reel must agree with that specified for the label of the file to be written.

The BSF routine cannot be used on a reel with mixed densities, since difficulty will be encountered in backspacing over files of different densities; nor can it be used to reposition in front of the data of a labeled file for which label and file have different densities.

It is strongly recommended that density changes be avoided.

#### Mode

Normally, the body of a file will contain information recorded in only one mode, binary or BCD. However, IOCS is capable of processing a mixed mode file whose blocks contain the standard look-ahead words as described in the IBM 7090 Operators Guide (Form A22-6535). The user need only describe this file as a mixed mode file on the Preprocessor FILE card and IOCS will determine the mode in which to read each block, although no indication will be given the user as to what that mode was.

A mixed mode output file can be written, but it is the user's responsibility to place the proper look-ahead words into the buffer. Iocs will always write the first block in the mode specified on the FILE card.

# Section 12—Printing Messages On-Line

IOEX provides a routine which facilitates printing messages on the Channel A printer, with editorial flexibility achieved by (a) composing the line from any number of designated parts, and (b) providing for sense printer instructions between lines.

The calling sequence is:

F* FOR REMA	RKS	
LOCATION	OPERATION	ADDRESS, TAG, DECREMENT/COUNT CO
1":		
112 67	8 14	15, 16
	TS.X.	MWR. 4
	PZ.E.	N.,.,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,
	P	L <sub>1</sub> , T <sub>1</sub> , M1,+5,1,2,*S,P <sub>1</sub> R <sub>1</sub> ,
	P	L, 2, T, 2, M2, +, 5, 1, 2, *, S, P, R, 2
	•	· · · · · · · · · · · · · · · · · · ·
	• • • • • •	
	P	L, N, T, N, M, N + 5, 1, 2, *S, P, R, N

The quantities N, P, L, T, M, and SPR are parameters which serve the following purposes:

- 1. N is the number of words in the calling sequence (following this one).
- 2. P, L, T, M. The M words (of 6 BCD characters each) beginning at location L, T are converted to Hollerith and placed in the line image for printing. If P = PZE, the image is taken to be complete and the line is printed. If P = MZE, this line is considered incomplete, and the L, T and M in the next word of the calling sequence are used in order to continue building the image.
- 3. SPR. If P = PZE, then the sense exit SPR is activated after the line is printed. (An SPR appearing in a word with P = MZE is ignored). Activation of an exit before printing the first line can be accomplished by inserting into the calling sequence:

-1	-* FOR REM	AΝ	RKS		
Ш	LOCATION		OPERATION	ADDRESS, TAG, DECREMENT/COUNT	င႖
	2 6	7	8 14	  15  16	_]
			P.Z.E.	*,*,,,,5,1,2,*,SP.R.	]
L					$\Gamma$

This will print a blank line, followed by activation of the sense printer hub, spr.

Suppose, for example, that in three consecutive words at location L1 we have the BCD text

THIS IS ONE

and at location L2 we have

SAMPLE

Then, assuming the share standard sense exit hubs, the sequence

_ * FOR RE	MΑ	RKS		
LOCATION		OPERATION		ADDRESS, TAG, DECREMENT/COUNT CO
1 2 6	7	8 14	15	16
1,,,,,		TSX.		MW.R., 4
1		$P_1Z_1E_1$		3
		$P_iZ_iE_i$	<u> </u>	$0_1, \dots, 5, 1, 2, \dots, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1,$
		MZE		L <sub>1</sub> ,,,3,
		$P_1Z_1E_1$		L21,+5,1,2,*4, , , , , , , , , , , , , , , , , , ,
11				

would result in

- (1) eject to a new page;
- (2) the line

#### THIS IS ONE SAMPLE

(3) a double space.

On the other hand, the sequence

_	.г.	-* FOR REM	AΑ	RKS		
		LOCATION		OPERATION		ADDRESS, TAG, DECREMENT/COUNT CO
	L	2 6	7	8 14	15	16
				TSX.		M,W,R, , 4
				P, Z, E, , ,	!	2, , , , , , , , , , , , , , , , , , ,
		1 1 1 1		P.Z.E.	<u> </u>	$L_{1}, \ldots, 3$
				PZE	<u> </u>	L <sub>1</sub> 2 <sub>1</sub> , 1, 1, 1, 5, 1, 2, * , 1

would result in

(1) the two lines

THIS IS ONE SAMPLE

(2) followed by a page eject.

The entry MWR is directed to the communication vector of IOEX in Basic Monitor. MWR points to IOEX +4. (See Basic Monitor bulletin entitled: IBM 7090 Operating Systems: Basic Monitor (IBSYS).)

# Section 13—Program Preparation

From a programmer's point of view, rocs is a collection of subroutines which perform a variety of file processing functions. In order to furnish a convenient standard procedure for using these subroutines, rocs has been incorporated into an operating package.

Briefly described, the operation of the system for any job consists of the following steps:

- 1. The iocs subroutines and the Preprocessor are loaded from a Basic Monitor System Tape.
- 2. The Preprocessor reads a set of control cards from the Program Input Unit, sysini, and creates, in storage, a group of File Control Blocks.
- 3. A loading sequence is initiated for the Program Input Unit to call the object program.

### **Storage Conventions**

The IOCS subroutines are assembled as a group and are located in core storage immediately following IOEX of IBSYS, Basic Monitor. All of storage beyond the end of IOCS is available for the object program, together with its working storage, buffers, and the File Control Blocks.

# References to System Subroutines

Within his program, a programmer must define the transfer points to those rocs subroutines which he uses. These transfer points are located relative to the origin of rocs itself:

FOR REM	ARKS	
LOCATION	OPERATION	ADDRESS, TAG, DECREMENT/COUNT CO
1 2 6 7	7 8 14	115 16
LOCS	EQU	L
DEFI NE	EQ.U	I OCS +4
JOI N	EQ,U,	1,O,C,S,+,6, , , , , , , , , , , , , , , , , ,
ATTACH	E.Q.U.	I O.C.S. +8.
CL OS E	EQU	1.0,C,S,+,1,0, , , , , , , , , , , , , , , , , ,
OPEN	E,Q,U	I O CS + 1 2
R E A D	EQ.U.	I,OCS, +1,4
WR, I,T,E	EQU,	I,OCS,+,16
COPY	EQU,	I O CS + 18
R EW	EQ.U.	1,OCS,+20, , , , , , , , , , , , , , , , , , ,
W.E.F.	EQ.U.	I,O,C,S,+,2,2,
BSR	E,Q,U,	I,O,C,S,+,2,4
BS F	EQ.U,	I,O,C,S, +2.6
CK PT	EQ,U	IOCS+28
S.T.A.S.H.	EQU.	I,O,C,S,+3,0
MWR	EQU,	', I,O,C,S, +3β
		_

The actual location corresponding to symbolic location rocs may vary from installation to installation. For any installation, however, this location and those

defined relative to it, as shown above, can be expected to remain fixed, even when modifications are made to the iocs system itself.

In addition to the above subroutine transfer points, certain other quantities which may be of interest to the programmer are located near the beginning of iocs:

Location Symbol	Position Relative to Location IOCS	Contents
DATE	IOCS + 39	System date. The internal mode is BCD, and the cell is of the form YYbXXX, where
		YY = Year XXX = Day of the year (001-365)
JOBID	IOCS + 40	
	$\left. egin{array}{c}  ext{to} \  ext{IOCS} + 42 \end{array}  ight\}$	Job name
LAREA	IOCS + 43	
	$\left. egin{array}{c}  ext{to} \\  ext{IOCS} + 56 \end{array} \right\}$	Label area (14 words)
LFLBLK	IOCS + 57	This word defines the origin and length of the File Block prepared by the Preproc- essor. It is of the form
		MZE A, , B where
		A= File Block origin
		B = File Block length
CPFILE	IOCS + 58	Address: Current checkpoint file in use
CPSEQ	IOCS + 59	1 + sequence number of last checkpoint written
LTSX	IOCS + 60	Prefix:
		0 – No error
		1 – Block sequencing error 2 – Check sum error
		4 – End of buffer error
		Decrement:
		2's complement of the quantity (1 + the location of the IOCS command on which the error, if any, occurred)
		Address:  2's complement of the location of the TSX instruction which last entered IOCS
XEOB	IOCS + 61	Decrement:  Location of the EOB exit
XEOF	IOCS + 62	Address: Location of the EOF exit

Location Symbol	Position Relative to Location IOCS	Contents
XERR	IOCS + 63	Decrement:  Location of the error exit
LTRAD	IOCS + 64	The last history word loaded into the AC: Prefix: PZE
		Decrement: $1 + \text{the location of the last word processed}$
		Address: Count of words remaining in the last buffer used
TRANS	IOCS + 65	Address:  Location of the first word processed by the last IOCS command
WDCT	IOCS + 66	Decrement: Word count of last IOCS command executed
IRS	IOCS + 67	Address: Contents of index register 2 at last entry to IOCS
		Decrement: Contents of index register 1 at last entry to IOCS
SENSE	IOCS + 68	Contents of Sense Indicators at last entry to IOCS.
FCW	IOCS + 69	Address: If = 0, IOCS was not in control If $\neq$ 0, this is the current file being used

# **Completed Program Return Point**

At the completion of a program, the standard action is a transfer to symbolic location IOCS. The system will then print a message indicating that the job has been completed, will set any reserve unit status for intersystem use, and will resume scanning for IOCS control cards on the Basic Monitor Input Unit.

#### **File Control Blocks**

Each 10cs subroutine calling sequence permits reference to one or more files. A file is specified in a calling sequence by giving the first location of the twelve word File Control Block for that file. A File Control Block contains certain fixed information about the file, such as the file type (input, output, or checkpoint), the recording mode and density, labeling conventions and the file name, as well as the assignment of input/output units. Other portions of this block are used by 10cs to post the current status and activity of the file.

The File Control Blocks required by a program are normally generated by the Preprocessor, just prior to execution of the program, from a set of control cards supplied with the program. In his program, the programmer need only provide the set of block reservations defining the files for program reference:

* FOR R	EMA	RKS			
LOCATION		OPERATION	!	ADDRESS, TAG, DECREMENT/COUNT	cd
<b>†</b>	ļ		į		)
1 i 2	6 7	8 14	4 i 15	16	(
FILE1		BSS.	L	1.2	(ب
F LLE2		BSS.	1	1, 2	4
		10 1 1 1 1	1	 	\ 
			-	 	(
		10 1 1 1 1	!	 	لب
FLEK		B <sub>i</sub> S <sub>1</sub> S <sub>1</sub>	-	; <mark>:1,2, , , , , , , , , , , , , , , , , , ,</mark>	السا
					<del>.</del>

#### File Block

The File Control Block reservations should be programmed, as shown above, to occupy a single storage area consisting of a succession of twelve word blocks. This area will be referred to as the  $File\ Block$ . It is, then, possible to assign to each of the File Control Blocks, in order, the numbers 1, 2, . . ., so that, given the origin of the File Block and the number of any File Control Block, the location of that File Control Block can be easily computed. If L is the File Block origin, the origin of File Control Block K is L+12(K-1).

#### **Unit Control Blocks**

The assignment of an input/output unit to a file is accomplished indirectly within a File Control Block by giving the location of a four-word Unit Control Block. It is in this Unit Control Block that the actual unit address is found.

The Unit Control Blocks, one for each I/O unit attached to the computer, are located in the Basic Monitor. Within the table, the Unit Control Blocks are grouped in *logical channels*, where a logical channel consists of the blocks for all I/O units attached to one machine Data Channel. For convenience of reference, the first logical channel is designated Channel A, the second is designated Channel B, and so forth. (Up to eight logical channels are permitted.) The Unit Control Blocks for all *tape* units of a logical channel occur together and are referred to, in order, by number  $(1, 2, \ldots)$ . Thus B3 refers to the third of the group of Unit Control Blocks of Logical Channel B which are assigned to tape units.

If, in addition to tapes, a logical channel contains one or more units of card equipment, these units are designated as RDX, PRX, and PUX for the reader, printer, and punch, respectively, on Channel x.

A Unit Control Block contains, besides the corresponding 1/0 unit address, the location of the first of any waiting 1/0 requests made for that unit. If the Unit Control Block is for a tape unit, it also contains counts which specify the current position of the tape (file count, and record count within the file).

## **Configuration Options**

Any one of four distinct configurations of the rocs system may be used by a given program, depending on its requirements. These configurations are listed below (in order of decreasing generality and core storage requirements):

- 1. Label 10Cs—This configuration comprises the complete 10Cs system. All of the described features are available; in particular, the labeling and label checking routines.
- 2. Basic 10cs-This configuration contains all of the 10cs features except the labeling routines.
- 3. Minimum 10cs—This configuration deletes the following routines and features from the Basic rocs:

JOIN	STASH	WEF	BSF
COPY	REW	BSR	CKPT

4. Executor—This option consists of the trap supervisor and the iocs communication cells.

#### **Control Cards**

Seven kinds of control cards are recognized by the Preprocessor (IOCS Monitor): JOB, FILE, DATE, LOAD, RESTART, IBSYS, and ID. The LOAD card is interpreted by the Preprocessor as the end of a control card set.

The job card, besides supplying the job name, defines the length of the File Block and its origin in core storage, and the iocs configuration for the program. If no File Block origin is given, a "standard" assignment is made by the Preprocessor. This standard origin is the cell following the end of rocs, and depends on the system configuration specified.

Each file to be processed is represented by a FILE control card. This card contains all of the information needed by the Preprocessor to generate a File Control Block. Space is provided on the card for a File Number which defines the relative location of the File Control Block within the File Block. The order of FILE cards in the control card set is immaterial.

The date card provides locs with a creation date for any labeled output files produced by the program. If this card is absent, the date is taken from the Basic Monitor.

This card initiates loading of the main program. It must appear as the last control card and may contain a Basic Monitor System Unit in its variable field.

This card initiates a restart from some previous checkpoint. It is equivalent to setting Sense Switch 2 down.

The IBSYS card transfers control to the Basic Monitor.

The id card can contain installation accounting information.

JOB CARD

FILE CARD

DATE CARD

LOAD CARD

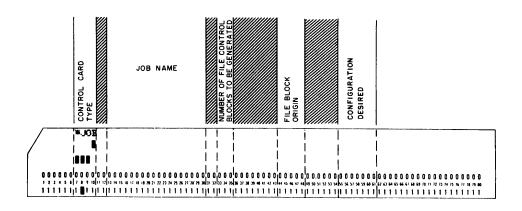
RESTART CARD

IBSYS CARD

ID CARD

#### **Control Card Formats**

JOB CARD



Columns	Contents	
7–10	*JOB	
13–30	Job name	
33–35	The number of File Control Blocks (N) in the File Block to be generated by the Preprocessor.	
44-48	File Block Origin (octal). If the standard origin is used, this field should be blank.  Desired configuration for this job:  MINIMUM—Minimum 10CS BASIC—Basic 10CS LABELS or blank—Label 10CS EXECUTOR—Basic Monitor Executor only.	
55–61		
1 🙀		

FILE CARD

CONTROL CARD TYPE MINIMUM FILE NUMBER MINIMUM	SECONDARY INPUT UNIT  SECONDARY INPUT UNIT  UNIT  SECONDARY INPUT  SECONDA	RETENTION DAYS MINISTRUCTURE MAY A PAIN A PA
PELLEI III	i iliiliiliiii ii	
		ii ii ii l
	1 11111111111   1	
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0	0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0	0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
111111111111111111111111111111111111111	ունու անիկիկիկիկիկիկին ու որ հի ա	144111441111111111111111

7–11	*FILE
13–15	This number indicates the relative position of the File Control Block within the File Block. It must not be greater than the number punched in columns 33–35 of the JOB card.
17	Tape mounting indicator:  * — the file is to be mounted immediately, i.e., when starting processing.  blank—this file is not required when processing is begun.
18–21	UNIT1. This is the symbolic designation within the table of Unit Control Blocks, of the primary input/output unit for the file. If the primary unit is a tape, this field is in the form xn, where x is the logical channel (A through H) and N a relative tape number. If the primary unit is a card reader, card punch, or printer, this field is in one of the following forms:  RDX—card reader  PUX—card punch  PRX—printer  where x is the logical channel. If the file is an Internal file, this field should contain the designation "INT."
22–25	UNIT2. This is the symbolic designation within the table of Unit Control Blocks, of the secondary unit for the file. The

**Contents** 

NOTE:

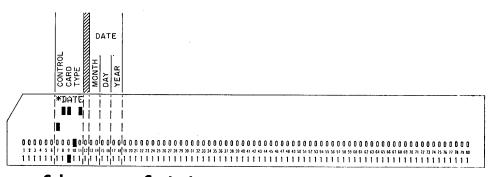
**Columns** 

UNIT1 and UNIT2 fields have special tape assignment procedures to facilitate inter-system communication. For complete information about these procedures, refer to the bulletin entitled *IBM 7090 Operating Systems: Basic Monitor (IBSYS)*.

format of this field is the same as for columns 18-21.

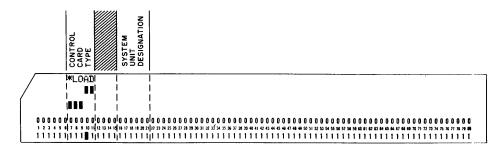
Columns	Contents		
27	List control N—Do not include this file in the Preprocessor File List blank—Include this file in the File List		
28	File type: C—Checkpoint I—Input T—Total Block Output P—Partial Block Output		
29	Reel control flag:  M—Multi-reel file, unlabeled file only  L—Search for label on open, labeled file only  blank—Single reel file, if unlabeled; no label search, if  labeled		
30	File density H—High density L—Low density		
31	File mode: D—BCD B—Binary M—Mixed mode, first record BCD N—Mixed mode, first record binary		
32	Labeling conventions: H—High density label L—Low density label S—Standard density convention blank—No label		
33	Block sequence numbering flag (significant for binary files only): S—Sequencing words provided blank—No sequencing words provided		
34	Check sum flag (available for binary sequence numbered files only):  C—Check sums included blank—No check sums included		
35	Checkpoint conventions:  F-Write checkpoints on this file at each reel switch (labeled file only)  C-Write checkpoints on the checkpoint file at each reel switch blank-No checkpoints initiated by this file		
36	Restart repositioning flag: N—No repositioning on restart blank—Repositioning on restart		
38–41	Reel sequence number of first reel to be processed in this file (significant for labeled files only)		
44-48	File serial number (meaningful for labeled input files only)		
51–53	Number of days file is to be retained		
55–72	File name		

#### DATE CARD



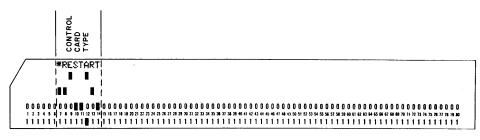
Columns	Contents
7–11	*DATE
13–14	Month (e.g., 06, 11)
15–16	Day (e.g., 03, 21)
17_18	Vear

LOAD CARD



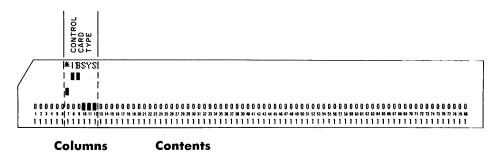
# 7-11 \*LOAD System Unit (e.g., SYSIN2) If this field is blank, SYSIN1 is assumed.

RESTART CARD



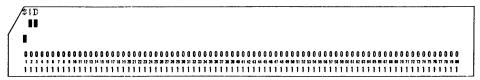
Columns	Contents	
7–14	*RESTART	

IBSYS CARD

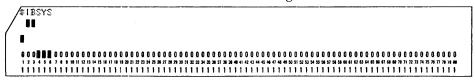


7–12 \*IBSYS

The Preprocessor will, in addition, recognize the following Basic Monitor control cards:



This card can be used for installation accounting.



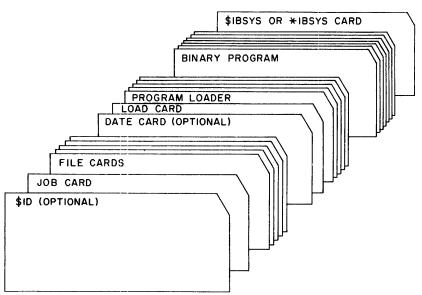
This card transfers control to the Basic Monitor.

#### Note:

\$IBSYS and \*IBSYS perform the same function. \*IBSYS is a Preprocessor control card; \$IBSYS is a Basic Monitor control card.

### **Program Input**

The program input for iocs consists of the following parts arranged in the order shown:



#### **Program Loading**

In order that the rocs system will be as independent as possible of the assembly and program loading techniques used by an installation, the Preprocessor contains no specific program loading facilities. Instead, after the control card set has been read and processed, the Preprocessor merely executes a standard loading sequence for the Program Input Unit (all Data Channel Traps are disabled at this time) designated by the \*LOAD card.

- 1. The loader must not destroy the contents of that portion of lower storage occupied by the Basic Monitor or by 10cs itself.
- 2. No portion of the File Block should be destroyed during loading.

# Part II:

# **Example of the Use of IOCS**

Following is a sample data processing problem: preparing an updated master file incorporating all changes (including additions to, and deletions from the file). This section will show some of the ways locs can be used to handle the input/output programming for the problem.

#### Input

Assume that one input file is the master file. This file is on tape in the binary mode and consists of 5-word records for various accounts; there are 100 records per block. An additional word is carried in each block; this word contains a block check sum and a sequence number. Each account on the file consists of a master record in the format:

Word	Contents	Internal mode
1	Account Number	Binary
2	Account balance	Binary
3	Year-to-date dividend	Binary
4	Year-to-date tax	Binary
5	Code (0)	Binary

Each master record is followed by five detail records, with code numbers 1-5. These records have the format:

Word	Contents	Internal mode
1	Account number	Binary
2–4	Description	BCD
5	First six bits: Code	Binary
5	Last 30 bits: Detail balance	Binary

The other input is the transaction tape, which is written in the BCD mode. There are 14 words per record, one record per block. Transaction records have codes 0 and 1-5 corresponding to the master and detail records. In addition, there is a code (7) for a deletion record. Type 0 transactions have the format:

Word	Contents	Internal mode
1	Account number	BCD
2	First 12 bits: Account number (con't)	BCD
2	Last 24 bits: Code (0)	BCD
3	12 bits each: Month-day-year	BCD
4	Old balance	BCD
5	First 12 bits: Old balance (con't)	BCD
5	Last 24 bits: Blank	BCD
6	Balance change	BCD
7	Dividend change	BCD
8	Tax change	BCD
9–14	Not used	

The other transaction types have similar format.

### Output

Output will consist of an updated master file; an error file, consisting of master records after conversion and transaction records in error; and a history file of all deletions and all type 0 transactions processed. The last two files are in the format of the transaction tape.

# **Processing**

The processing steps are:

- 1. Compare transaction and old master file by account number.
- 2. Write unmatched masters without change.
- 3. Create a new master for unmatched transactions.
- 4. Process matched conditions by type, as follows:
  - a. For type 0, the old balance of the transaction is compared with the account balance of the master. If the balances are different, the master is converted to BCD and written on the error file, followed by all transactions. If the balances are the same, the new balance is posted.
  - b. Types 1-5 follow in order. There may be multiple transactions of each type; if so, the date should increase. Update each detail type.
  - c. On a type 7, the old balance must again match the account balance in the master. If it does not match, the account is written on the error tape along with the transaction. If it does, it is deleted from the master file and written on the history file.

# **Programming the Input/Output**

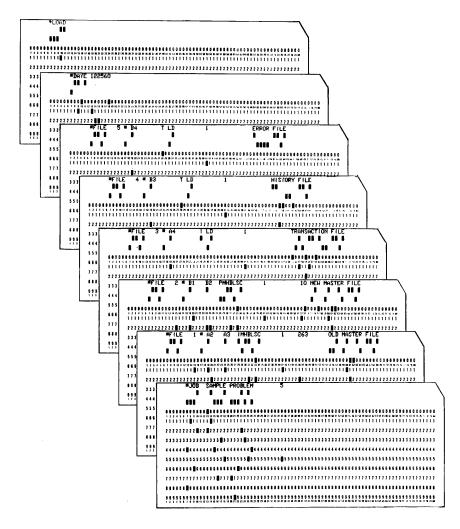
Considerations which arise in using rocs for the input/output of the sample problem are discussed below.

# **File Specifications**

As the file specifications involved in the problem have no effect on the programming, it will simply be assumed, for purposes of discussion, that the File Control Blocks are defined as follows:

MAITRE	BSS	12	Old master file
NEWMR	BSS	12	New master file
ERF	BSS	12	Error file
HIST	BSS	12	History file
TRANS	BSS	12	Transaction file

The following control cards might be used:



#### Note

References to actual units are interpreted as relative references by the Basic Monitor. The \*LOAD card may have a particular unit designated in columns 16-21. If no unit is specified, the Preprocessor will initiate loading from SYSIN1.

The five files fall naturally into two groups: the two master files, which have large block size, and the other three files with small block size. Because of the disparity in block size of the two groups, it is desirable to use two buffer pools. Suppose these to be located at BIGBUF and LILBUF, and that sufficient core storage space is available. The first instruction sequence is:

* FOR RE	MΑ	RKS		
LOCATION		OPERATION	ADDRESS, TAG, DECREMENT/COUNT	COMMENTS
1 2 6	7	8 14	5,16	
		T.S.X.	D.E. F. I.N.E., 4. DEFINE BIGBUF POOL	AS TEN
		P.Z.E.	B.I. G.B.U.F. 501-WORD BUFFERS	
		$P_iZ_iE_i$	1,0,,,,5,0,1,	
1		T.S.X.	DEFINE LILBUF POOL	AS TEN
		P.Z.E.	L,I,L,B,U,F, 12-WORD BUFFERS	
1		P <sub>i</sub> Z <sub>i</sub> E	1,0,,,,12,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,	

501 words are needed for each buffer in the BIGBUF pool because the master files have block check sums and sequence numbers. (This extra word will be auto-

# **Buffer Pools**

matically bypassed by the system routines, and the programmer will never be aware of it again unless an error occurs.) Specifying ten buffers for each pool is, of course, arbitrary and subject to space limitation. There are 5032 cells in the bigbuf pool, and 142 cells in the lilbur pool.

#### Initialization

The following list (actually two separate lists) is placed in storage:

_	* FOR REI	* FOR REMARKS					
Į,	LOCATION		OPERATION		ADDRESS, TAG, DECREMENT/COUNT	COMMENTS	
	2 6	7	8 14	18	18		
F	LIST.1		P.Z.E.	_	NE.WMR	NEW MASTER FILE IS IMMEDIATE	
-	<del>                                     </del>		SVN	L	1,,,2,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,	OLD MASTER FILE IS A RESERVE	
L	<del> </del>		PZE	<u>.                                    </u>	MAITRE	FILE HAVING TWO BUFFERS.	
L	LIST 2		P <sub>i</sub> ZE <sub>1</sub> 1		r.R.A.N.S.	TRANSACTION, HISTORY, AND	
L	1 1 1 1 1		PZE		IJ S.T.	ERROR FILES ARE ALL IMMEDIATE.	
-	<del> </del>		PiZiE		E.R.F.		
L	لىبىل			L		<del></del>	

The following initialization sequence is then used.

		_		•			
FOR REM	FOR REMARKS						
LOCATION		OPERATION		ADDRESS, TAG, DECREMENT/COUNT COMM	MENTS		
1 2 6	7	8 14	15	16			
		T,S,X, , ,		A,T,T,A,C,H,,4, ATTACH MASTER FILES, A	ND		
		P.Z.E.	_	B.I.G.B.U.F. OPEN NEW MASTER,			
		P <sub>i</sub> Z <sub>i</sub> E <sub>1</sub>	<u> </u>	L <sub>1</sub> I <sub>2</sub> S <sub>1</sub> T <sub>1</sub> I <sub>1</sub> , 1, 2, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1,	<u> </u>		
		T.S.X.		ATTACH AND OPEN TRANS			
		P.Z.E.		L.I.L.B.U.F. ACTION, HISTORY, AND			
1		P.ZE.		L,I,S,T,2,3 ERROR FILES.	1 1 1 1-		
		T.S.X.		O.P.E.N., 4. OPEN OLD MASTER FILE.	ــــــــــــــــــــــــــــــــــــــ		
		P.Z.E.		MAITRE	والمسلسلة		
			L	 			

The old master is treated as a reserve file in order to assure that two buffers are always available for it. Two buffers are necessary, since each block has 100 records, which is not a multiple of the number of records (6) per account, so that during the reading process (see below) the records may be in different buffers.

#### **Processing Master Files**

The old master file is read by the sequence:

FOR REM	* FOR REMARKS						
LOCATION		OPERATION	1	ADDRESS, TAG, DECREMENT/COUNT	COMMENTS		
1 2 6	7	8 14	15	ı B			
	Г		1	READ OLD MASTER	FILE		
<del></del>		T.S.X.	H	R.E.A.D., 4	ــــــــــــــــــــــــــــــــــــــ		
		P.Z.E.		MALTRE, PROSS LOCATING:			
		P.Z.E.		NION E.,, TILT			
1.0.1		I,O,C,P,N,		* * MASTER ACCOUNT 1	RECORD		
1.0.2		IO.C.P.N.		*,*,.,5	CODE 1		
1.0.3		IOCP.N.		*,*,,,5, ACCOUNT DETAIL,	CODE 2		
104		IO C.P.N.		*,*,,,5	CODE 3		
1.0.5		IO.C.P.N	! !	*,*,,,5,	CODE 4		
1.0.6		IO.CT.N.		*,*,,,5,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,	CODE 5		
PROSS		LA.C.		BEGIN PROCESSING	WITH		
			1	MASTER ACCOUNT 1	RECORD.		
			1 1		1 1 1 1 1 1 1		

When PROSS is reached, each of the commands 101-106 will have the address of the first word of the corresponding record type for the current account.

When the account group is to be written on the new master file, all that is needed is:

* FOR REM	ARKS	
LOCATION	OPERATION ADDRESS, TAG, DECREMENT/COUNT	COMMENTS
1 2 6	7 8 14 15 16	
	T.S.X. WRITE ON NEW MAST	ER, TRANS-
	P.Z.E. N.E.WMR MITTING EACH RECO	RD FROM ITS
OU.T.1.	IO.C.P.* LOCATION IN THE INI	PUT BUFFERS.
	I.O.C.P.*	
	I.O.C.P.* I O.3., 5	
	I.O.C.P.*. I.O.4	1 1 1 1 1 1 1
	I.O.C.P.*	
	I.O.C.T.* I O.6., 5	
لببيا		

Since the indirect effective address is computed without index registers, the non-transmitting bits in the tags of the commands 101-106 have no effect.

### **Processing Transaction Files**

The transaction file is read by the sequence:

FOR REMA	ARKS		
LOCATION	OPERATION	ADDRESS, TAG, DECREMENT/COUNT	COMMENTS
1 2 6 7	8 14	15 16	
	T.S.X.	READ TRANSACT	ION FILE TRANS-
	P.Z.E.	TRANS,,, W3 +1 MITTING ACCOU	NT NUMBER
	P.Z.E.	NOMORE REDUN AND CODE TO W	ORKING
	I <sub>1</sub> O <sub>1</sub> C <sub>1</sub> P <sub>1</sub>	WORD1	COCATING
W. 3.	I.O.R.T.N.	* * THE REMAINDER	OF THE RECORD.
لسسلا	L		

Since the sequence ends with a non-transmitting buffer control command, the entire transaction record is available in the buffer. In addition, the account number and type code are in cells word and word +1, ready for some conversion routine. The address of the command in location w3 has the buffer location of the data fields for this transaction.

Since the transaction, error, and history files are attached to the same pool, any transaction which has been read by the sequence above can be written by:

_	* FOR RE	MA	RKS	
1	LOCATION		OPERATION	ADDRESS, TAG, DECREMENT/COUNT CC
L	2 6	7	8 14	15 16
L	<u> </u>		T, S,X,	C,O,P,Y,,,4
F	<u> </u>		P <sub>i</sub> Z <sub>i</sub> E <sub>1</sub> , ,	T.R.A.N.S.,, H.I.S.T. (10.r. E.R. F.)
L	<del></del>	l		<del></del>

# **End of File: Transaction File**

When the transaction tape is depleted of records, the READ sequence will exit to the routine at location NOMORE. That routine will handle the case of an end of file on the transaction file before the old master has reached end of file; the routine at NONE will handle the case where end of file occurs first on the old master file. In the first situation, processing is finished on the files TRANS, ERF, and HIST,

and the problem is completed by copying the rest of the old master onto the new. The completion is carried out by:

-* FOR REMA	RKS			
LOCATION	OPERATION	1	ADDRESS, TAG, DECREMENT/COUNT	COMMENTS (
1 2 6 7	8 14	15	16	
NOMORE	T.S.X.	1	CLOSE TRANSA	CTION ERROR,
	P.Z.E.	ī	L.I.S.T.23	TLES WITH
*		-	REWIND-U	NLOAD
	m a **	<del> </del>	PUT OUT THE	CURRENT
	T <sub>i</sub> S <sub>i</sub> X <sub>i</sub>	Ī	ACCOUNT RECO	/
	P.Z.E.	ī	(N) 15 (W) 14 15 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1	
	T,C,H,	-	LOCATE WORD	<u> </u>
FIND	T.S.X.	ļ.,	$R_1E_1A_2D_1$ , $14$	<del>/</del>
	P.Z.E.	<u>i</u>	MAITRE, DUPE+1 REMAINING IN	READ
	P.Z.E.	į.	END, TILT	
DU.P.E.	I ORTN	L	.*. * <sub> •</sub>  • * *, ]	
	$C_1L_1A_1$	!	SET UP WRITE	SEQUENCE
	S <sub>1</sub> T <sub>2</sub> D <sub>1</sub>	Ì	F.I.L.L.	, -4111111111111
		亡		ORMATION FROM /
	T <sub>i</sub> S <sub>i</sub> X <sub>i</sub>	+	! INPUT TO OUT	PUT BUFFERS
	$P_iZ_iE_i$	į-	N.E.W.M.R	
FILL	I.O.C.T.*	-	<u>'DUPE, , , , * , * , , , , , , , , , , , , ,</u>	
	T <sub>1</sub> R <sub>1</sub> A <sub>1</sub>	1	F. I.N.D.	P FILES WATH
E N <sub>1</sub> D <sub>1</sub>	$T_iS_iX_i$	<u>i</u>	CLOSE MASTE	
	P.Z.E.	i.	LIST1,,,2	AD
	1	-		<u> </u>

The first write, at NOMORE +2, writes the account records which had previously been located. The loop from find to fill+1 copies the remainder of the old master file onto the new master file. When an end of file is reached on the old master file, the close at end writes the trailer label and rewinds and unloads the files. This use of record control for the read location and count control for the write transmission completely eliminates the problem of adjusting for the fact that the old and new master file records may be in different positions within the block.

### End of File: Old Master File

The routine at NONE handles the case which arises when an end of file occurs on the old master before it does on the transaction tape. In this situation, new masters are to be created from the transaction tape until it is exhausted. Instructions to do this are:

-* FOR REMA	RKS		
LOCATION	OPERATION	ADDRESS, TAG, DECREMENT/COUNT	COMMENTS
]*:		j	
112 67	8 14 115	1	CLOSE OLD MASTER AS FULL
NONE.	T.S.X.	C,L,OSE,,4	
	P.Z.E.	L, I, S, T, 1, +, 1, , , , 1, , , , , , , , ,	RESERVE GROUP
	1	1	READ NEXT TRANSACTION
	T.S.X.	R.E.A.D 4	RECORD, LOCATING IN
	P.Z.E	T.R.A.N.S., *, +, +, 3.	
1	$P_iZ_iE_i$	F.I.NAL., RED.U.N	BUFFER
	1	1	CREATE MASTER FILE RECORDS
	I ORTN	<del> * * *                               </del>	TOD STATE A GROUNT GROUP IN WEAC
11	1	<u> </u>	FOR NEW ACCOUNT GROUP IN WSAG.

and (when the account group is complete):

FOR REM	ARKS	
LOCATION	OPERATION ADDRESS, TAG, DECREMENT/COUNT	COMMENTS
1 2 6 7	6 14,15,16	
	T.S.X. WRI T.E., 4. WRITE	OUT NEWLY CREATED
	P.Z.E. N.E.W.M.R. MASTER	FILE RECORDS.
	I.O.C.T. WS A.G.,,,3,0	
	TRA NONE +2	
F.I.N.A.I	TSX, CLOSE, 4, CLOSE	NEW MASTER FILE
	PZ.E. N.E.W.M.R.	
	TSX. CLOS E . 4 CLOSE	TRANSACTION, ERROR
	PZE LIST23 AND HIS	TORY FILES
		7

# Variable Length Records

To demonstrate the flexibility of the system, suppose that the master file records are of variable length. Assuming the convention that the first word of a variable length record is a flag word of the form:

_	* FOR REMARKS						
,	LOCATION	OPERATION ADDRESS, TAG, DECREMENT/COUNT	co				
L	2 6	3 14 15 16	- 1				
L	<del></del>	I.O.C.P.N. ** n.					
L	<u> </u>						

where n is the number of words in the remainder of the record, and assuming also that a record will not span a block, the master file is now read by:

LOCATION	OPERATION	ADDRESS, TAG, DECREMENT/COUNT COMM
1 2 6	7 8 14	15 16
	T.S.X.	R.E.A.D., 14.
+	PZE.	MAITRE, PROSS
<del>  </del>	P.Z.E.	NONE TILT
+	LOCIR	* <sub>+</sub> + <sub>1</sub> READ IN FLAG WORD AND
1.0.1	P.Z.E.	**,
+	I.OC.P.	*,+,1,,,,,1,,,,,,,,,,,,,,,,,,,,,,,,,,,,
1.0.2	PZE	*,*
	LOCIP	*+1.,1.
1.0.3	P.Z.E.	*,*, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1,
+	I.O.C.P.	*,+,1,,,,1,,,,,,,,,,,,,,,,,,,,,,,,,,,,,
1.0.4	PZE	*,*, , , , , , , , , , , , , , , , , ,
	I.O.C.P.	*+:1:::::::::::::::::::::::::::::::::::
1.0.5	PZE	  * *,
<del> </del>	I.O.C.P	!*,+, <u>1,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,</u>
I O.6	P.Z.E.	**
	LOCITIN	* * TERMINATE I/O LIST
PROSS	LAC.	I O 1 1
i		

The words at 101, 102, . . . , 106 have exactly the same information as for the fixed record case, plus the record lengths in their decrements. Hence, processing

can be basically the same. Writing is accomplished with special count control to insure that records do not cross a block gap.

Inasmuch as the program will terminate either by an end of file on the old master or on the transaction tape, the programming above is all that is needed to meet the input/output requirements of the problem, except for routines at TILT and REDUN to take action on error conditions. Such routines are subject to variation according to installation practices. Discussion is, therefore, omitted.

# Part III: IOCS Operation

### Section 1 — Control Sense Switches

Four control sense switches are used by iocs, the actual assignment of which is an installation choice. In this manual, the switches are referred to by their functional names.

# **Input Control**

Input control (Sense Switch 1) is a function of the Basic Monitor. See the bulletin entitled 7090 Operating System: Basic Monitor (IBSYS).

#### **Restart Control**

Up The Preprocessor assumes that a new job is being started.

Down The Preprocessor assumes that a job is to be restarted, using checkpoint information from tape.

# **Checkpoint Control**

This switch is tested before every exit from an iocs subroutine.

Up No action is taken.

*Down* A checkpoint is written, provided that a checkpoint file has been opened. Once a checkpoint has been written, this switch must be returned to the Up position and sensed there by rocs before it can again be placed Down to cause writing of another checkpoint.

#### **Label Control**

This switch is used by the operator to indicate which of two possible actions should be taken when a label error is detected.

Up When the Start key is depressed, the tape in error is rewound and unloaded. When a new reel has been mounted, the label check is repeated.Down When the Start key is depressed, the tape is accepted despite the error.

# **Error History Control**

If one or more tape redundancy errors (correctable or uncorrectable) occur while processing a tape reel, a summary of this information is printed when the reel changing instructions are given to the operator. If no redundancy errors are detected, this summary is normally not printed.

This sense switch makes it possible to print the redundancy error history of each tape reel processed, whether or not errors occurred.

Up Print error history only if one or more errors occurred.

Down Print error history for each reel.

The standard sense switch assignments are as follows:

Function	Sense Switch
Restart Conrtol	2
Checkpoint Control	3
Label Control	4
Error History Control	5

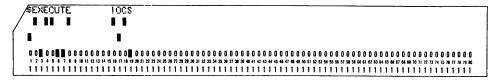
These assignments may be changed by re-assembly with appropriate changes in assembly parameters.

# Section 2—System Operating Notes

#### **Start Procedure**

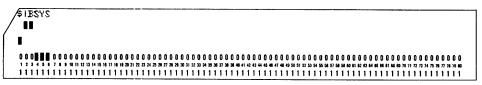
Since iocs is a participating system under Basic Monitor control, the iocs jobs are stacked on the Basic Monitor Input Function SYSIN1. All iocs jobs which are arranged in this manner are usually separated by EOFs on SYSIN1.

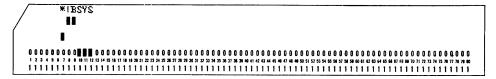
1. The first card of a stacked run is the IBSYS control card:



This card calls in the IOCS system monitor and transfers control to it.

2. At the completion of each IOCS job, scanning for the next job is resumed. At the end of all jobs, control should be transferred back to the Basic Monitor with one of these cards:





# System Operation

- 1. The control cards are read. If any errors are detected by the Preprocessor, an error list is printed.
- 2. The Preprocessor prints a file list.
- 3. If one or more tapes are to be mounted immediately, or if an error list was printed, one or both of the following messages may be printed:

#### MOUNT INDICATED TAPES

### CONTINUE IF CONTROL CARD ERRORS CAN BE IGNORED

If a halt occurs, the operator may depress the Start key, after any required tape mounting, to continue.

- 4. The object program is loaded, and control is transferred to it.
- 5. If the program takes the recommended action when the job is completed, the system informs the operator of this completion and calls the iocs monitor.
- 6. If an end of file is encountered while reading control cards, the message CONTROL CARD EOF

is printed.

PRESS START TO CONTINUE

7. Any error halts or illegal loops should be dealt with as follows:

First, execute a STR. This will preserve the loop location and will enter the Basic Monitor dump routine, SYSDMP. When the core dump is complete, control is transferred back to the Preprocessor (100s Monitor).

If, for some reason, the STR fails, enter SYSDMP directly by a transfer to location 115<sub>8</sub>. This will give the same results as the STR, except that the loop/halt location is not preserved.

## **Preprocessor Error List**

The Preprocessor prints an error list if one or more errors are detected while the control cards are being processed. Each error message in this list is of the general form:

CARD XXX (Error Condition)

For each error condition the Preprocessor takes some specific action to permit further card processing. A single halt occurs just prior to program loading if any errors have been found in the control cards.

### **Preprocessor Error Messages**

## ILLEGAL OPERATION –(XXXXXX)

Columns 7-12 of a control card contain XXXXXX which is not \*job, \*FILE, \*DATE, \*LOAD, \*RESTART, or \*IBSYS.

#### FILE BLOCK ORIGIN NOT OCTAL

The origin specified in the job card is ignored, and the standard File Block origin is used.

#### FILE CONTROL BLOCK COUNT ERROR

The value 50 is used for the count.

# FILE CONTROL BLOCK NUMBER ERROR

The field is either non-numeric or greater than the count given in the JOB card. No File Control Block is generated for this file.

#### UNIT1 ILLEGAL

The field does not contain a proper designation. No unit Control Block location is stored, and any reference to the file will cause an error halt.

#### UNIT2 ILLEGAL

If unit1 was correct, that designation is used for unit2.

#### DATE ERROR

The date from Basic Monitor is used.

#### JOB CARD OUT OF PLACE

The card is ignored.

#### CONTROL CARD REDUNDANCY

A redundancy has been detected while reading control cards from tape. The card is processed as read.

#### REEL SEQUENCE NUMBER NOT NUMERIC

The sequence number is assumed to be 1.

### **Preprocessor File List**

The Preprocessor prints a list of all files which the program is to use (see page 58). The following information is given for each file:

File number

File name

Physical I/O unit assigned to the file

Starting reel sequence number

If the file is an output file, what kind of tape to mount (labeled or unlabeled blank)

Whether the file must be mounted initially.

If two I/O units are assigned to a multi-reel file, the entry for that file consists of two lines, one for each unit.

For an output file, one of three mounting instructions may be given:

BLANK – UNLABELED
BLANK – LOW DENSITY LABEL
BLANK – HIGH DENSITY LABEL

A file which is to be mounted immediately is marked with an asterisk (\*) to the left of the line. Instructions for the mounting of files not so marked is deferred until the file is brought into use.

#### Sample File List

IOR -	· TEST	6		DATE	6/30/60	PAGE 1		
		_						
FILE DESCRIPTIONS (MOUNT FILES MARKED WITH *)								
	NO.	FILE	NAME	UNIT	MOUNT TAPE	S		
*	1	FILE	Α	Al	REEL 0001			
				A2	REEL 0002			
	2	FILE	В	А3	REEL 0002	BLANK-UNLABELLED		
	3	FILE	C	A5	REEL 0003	BLANK-LOW DENSITY LABEL		
				A6	REEL 0004	BLANK-LOW DENSITY LABEL		
*	5	FILE	D	A4	REEL 0001	BLANK-HIGH DENSITY LABEL		
	7	FILE	E	B1	REEL 0002			
	10	FILE	F	В2	REEL 0003	BLANK-LOW DENSITY LABEL		
	11	FILE	G	В3	REEL 0001	BLANK-HIGH DENSITY LABEL		
				B4	REEL 0002	BLANK-HIGH DENSITY LABEL		
*	12	FILE	н	85	REEL 0002	BLANK-LOW DENSITY LABEL		
	21	FILE	Κ	Al	REEL 0002	BLANK-LOW DENSITY LABEL		
	22	FILE	L	A2	REEL 0003	BLANK-LOW DENSITY LABEL		
				А3	REEL 0004	BLANK-LOW DENSITY LABEL		
	25	FILE	М	A4	REEL 0001			
*	26	FILE	N	A5	REEL 0002	BLANK-LOW DENSITY LABEL		
				A6	REEL 0003	BLANK-LOW DENSITY LABEL		

# **IOCS System Messages and Halts**

A number of messages are printed on the iocs system printer during the execution of a program. Some of these messages merely record the normal operation of the system. Others request some operator action, or describe an error condition which has been detected.

Associated with operator action and error messages, there are two procedures in iocs:

(1) A mnemonic core dump on sysoul and a return to the iocs monitor occur when some error condition is detected which makes it impossible to continue. The information printed at this time is of the form:

(Specific error message) CANNOT PROCEED

 $(2) \qquad \qquad \mathbf{HPR} \quad -\mathbf{1}$ 

(The Instruction Counter at this point is in the IOEX communication area.) This stop occurs when some operator action must be taken (e.g., tape mounting), or when a condition is detected within IOCS which may or may not be a significant error.

The message at this time can take one of two forms:

(a) If the operator is required to take some action, the following message is printed:

(specific operator action request)
OPERATOR ACTION PAUSE

When the Start key is depressed following this stop, the following line is printed:

#### ACTION COMPLETED

(b) If the halt occurred because of an error condition, the following message is printed:

(specific error message)

PRESS START TO CONTINUE

When the Start key is depressed following this stop, an additional line is printed:

# CONDITION IGNORED

TAPE MOUNTING AND REMOVAL MESSAGES IOCS assumes that those files marked for immediate mounting by the Preprocessor File List have been mounted. Thereafter, all necessary instructions to the operator to remove or mount tapes are printed by IOCS.

1. Mounting instructions are printed for any Immediate file when the file is attached, and for any reserve file when opened, except those indicated in the File List for immediate mounting:

UNIT XXXXX (file name) MOUNT  $\begin{cases} REEL \ YYYY \\ BLANK \\ LAB. \ BLANK \end{cases}$ 

Density of label can be determined from file list.

2. A history record and removal instructions are printed for each reel, of any file, which is rewound and unloaded:

UNIT XXXXX (file name) REEL XXXX - XXXXX RECORDS

REDUNDANCY HISTORY XXXXX RECOVERED XXXXX  $\{ERASE\}$ 

This message shows the number of redundancies corrected, and the number of erased areas when writing (ERASE) or permanent redundancies when reading (PERM.) The Error History Control Sense Switch can be used to force printing of this message when the error counts are zero. The message is always printed when the counts are non-zero.

UNIT XXXXX (file name) REMOVE REEL YYYY

- 3. When reel switching occurs for multiple reel files:
  - a. If a single 1/0 unit is assigned for the file, 10cs halts to permit the reels to be changed.
  - b. If two units are assigned, rocs does not halt, but automatically begins processing the second unit.

In both cases, the removal and mounting messages are printed.

NORMAL LABEL MESSAGES: Output files

If a labeled output file is to be prepared, the blank tape on which the file is to be written must possess a label when mounted.

The labeled tape can be obtained in two ways:

- 1. By using a tape on which some labeled file has already been written, if the tape need no longer be retained (this requires that the file Creation Date + Retention Days be less than the current date).
- 2. By creating a dummy label on the tape by an off-line card-to-tape operation.

When a labeled output file is opened, the following message is printed to indicate the contents of the label:

Input files

When an input file is opened, a label is read and compared with the information given on the corresponding file card. If a disparity is detected and the file card does not specify searching, the INPUT REEL INVALID error occurs (see label error messages). However, if searching occurs, each label encountered is printed on-line until the desired file is found. The complete message is:

(file name) SEARCH SKIP LIST FOLLOWS IF SSWX DOWN REMOUNT (label of file skipped)

(label of file skipped)

(file name) FOUND

Reel switching may occur automatically during the search. Also, if the specified sense switch is depressed during the search, a rewind and unload is issued and the search will continue on the next reel mounted on that tape drive.

LABEL ERROR MESSAGES

UNIT XXXXX (file name) LABEL ERROR (label)
(cause of error – see below)
IF SSWX DOWN IGNORED
OPERATOR ACTION PAUSE

causes of error:

#### a. INPUT REEL INVALID

Either the file serial number, the reel sequence number, or the file name is incorrect. If the Label Control Sense Switch (see page 55) is Down when the Start key is depressed, the reel is accepted as correct. If the switch is Up, the tape is rewound and unloaded. When a new tape is readied, it will be checked.

#### b. OUTPUT REEL INVALID

Reel not labeled (i.e., the label does not begin with "1HDR" or "1BLANK." If the Label Control Sense Switch is Down when the Start key is depressed, the dummy file serial number \*\*\*\*\* is used for the label. If the switch is Up, the reel is rewound and unloaded. Then when a new reel is loaded, the label will be checked.

#### c. RETENTION NOT EXPIRED

For tape xxxxx, (creation date + retention days) > (current date). If the Sense Switch is Down when the Start key is de-

pressed, the reel is used for output. If the switch is Up, the tape is rewound and unloaded. Then when a new reel is readied, it will be checked.

#### d. NO LABEL, BLANK CREATED

The record read from an output file as the label was not fourteen words in length. If the Label Control Sense Switch is Up, the tape is rewound when the Start key is depressed. When a new tape is readied, it will be checked. If the switch is Down, the system will proceed. (*Note:* Care should be used, since a lblank-type label is created, which may cause a subsequent error message.)

# MISCELLANEOUS LABEL ERROR MESSAGES

#### (file name) NO LABEL, BLANK CREATED

The record read as the label of an input file was not fourteen words in length. A lblank image will be created in the label area; this will probably result in a subsequent error message. There is no stop associated with this error.

#### (file name) NO TRAILER

The record read as the trailer label of an input file was not fourteen words in length. The end of file condition is assumed. There is no stop associated with this error.

# FILE PROCESSING ERROR MESSAGES

# POOL ERROR TSX AT XXXXX CANNOT PROCEED

An attempt has been made either: (1) to define a buffer pool using the storage area occupied by rocs; (2) to join a buffer pool to a non-existent pool; or (3) to join a pool to two different pools.

# ATTACH ERROR AT XXXXX CANNOT PROCEED

There are not enough buffers in the required pool to attach the specified files. xxxxx is the location of the TSX ATTACH, 4 instruction which caused the error.

# (file name) NOT AVAILABLE FOR INITIALIZATION PRESS START TO CONTINUE

An attempt has been made to attach a file which has already been attached to another pool or previously closed with a Rewind Unload. If the Start key is depressed, the program continues but the file is not opened.

# ILLEGAL FILE USE CANNOT PROCEED

This message indicates that a file designated as Total Block Output type is used for Partial Block Output.

# (file name) OPEN COUNT EXCEEDED—OPEN NOT POSSIBLE PRESS START TO CONTINUE

An attempt has been made to open too many files in a Reserve or Internal Group. If the Start key is depressed, the program continues without opening the file.

# ILLEGAL TRANSMIT $\{READ \\ WRITE\}$ XXXXX

#### CANNOT PROCEED

This error is caused by an attempt to read a word from or write a word into the area occupied by IOCS. XXXXXX is the location of the IOCS command which caused the error.

# ILLEGAL UNIT REQUEST/TRAP AT YYYYY

#### CANNOT PROCEED

YYYYY is either (1) the address of a select instruction which refers to an illegal or unavailable unit or (2) the location where the machine trapped for a non-existent channel.

#### TAPE ERROR MESSAGES

# UNIT XXXXX FILE XXXX REC. XXXX PERMANENT READ REDUN.

A specified number of consecutive redundancies have occurred while attempting to read an input record. The record is used as read the nth time. The number of times a check is made is determined by Basic Monitor Parameters.

# UNIT XXXXX FILE XXXX REC. XXXX 25 ERASES DURING WRITE

Twenty-five erasures have been made while attempting to write an output record. The program will continue to attempt writing.

### UNIT XXXXX FILE XXXX REC. XXXX NOISE RECORD DISCARDED

A redundancy check has occurred on the first attempt to read a tape record of two or less words. The record is considered to be tape noise and is ignored.

### UNIT XXXXX FILE XXXX REC. XXXX SHORT RECORD WRITTEN

A record of 1 or 2 words has been written. Note that if such a record is redundant when later read, it will be treated as a noise record.

### UNIT XXXXX FILE XXXX REC. XXXX NOISE ON ERASE

An attempt to erase a section of tape on which a record could not be written has produced a redundancy check.

#### UNIT XXXXX EOT ON ERASE

#### CANNOT PROCEED

End of tape encountered while trying to erase tape on which redundancy occurred.

#### I/O CHECK

#### PRESS START TO CONTINUE

A machine error has probably occurred. When the Start key is depressed, the message condition ignored is printed.

# CONFIGURATION ERROR MESSAGE

# BASIC IOCS NECESSARY XXXXX

or

# LABEL IOCS NECESSARY XXXXX

#### CANNOT PROCEED

The rocs configuration being used does not contain some routine required by the program. xxxxx is the octal location of the reference to the missing routine.

# Section 3—Checkpoint and Restart Procedures

# **Operator Initiated Checkpoints**

IOCS has provision for an operator to take a checkpoint if a checkpoint file has been assigned by the Preprocessor and the file has been opened by the program. This feature may be used to interrupt the execution of a program and to restart it at some later time. The procedure is as follows:

- 1. Set the Checkpoint Control Sense Switch Down.
- 2. The following message is printed:

CHECKPOINT XXXXX CODE YYYYYY ZZZZZZ

The checkpoint identification is xxxxx and the restart code for this checkpoint is YYYYYY ZZZZZZ. The physical tape address of the unit on which the checkpoint is taken is YYYYYY.

3. Restore the Checkpoint Control Sense Switch.

#### **Restart Procedure**

The following are operator procedures for restarting:

- 1. Obtain the identification number (xxxxx) and restart code (yyyyyy zzzzzzz) of the desired checkpoint record, which were printed at the time the checkpoint was taken.
- 2. Mount the tape which contains the checkpoint record on the tape unit which is indicated by the left half of the restart code (YYYYYY).
- 3. Set the Restart Control Sense Switch Down, or use \*RESTART Card.
- 4. \$EXECUTE IOCS via Basic Monitor operation.
- 5. After the message SET KEYS TO RESTART CODE is printed, a halt occurs. Enter the correct code (YYYYYY ZZZZZZ) in the panel entry keys and depress the Start key.
- 6. The checkpoint identification is printed: RESTART XXXXX. Verify the identification number.
- 7. The following message is printed:

SET SENSE SWITCHES 
$$1 {UP \ DN} 2 {UP \ DN} 3 {UP \ DN}$$

$$4 {UP \ DN} 5 {UP \ DN} 6 {UP \ DN}$$

Set sense switches accordingly.

8. A summary of tape usage is printed and a halt occurs again. This summary is similar to the File List printed at the beginning of a job. Mount tapes as indicated, and then depress the Start key.

#### **Messages During Restarting**

RESTART XXXXX

SET SENSE SWITCHES 1XX 2XX 3XX 4XX 5XX 6XX
(Restart message – any number of lines)

OPERATOR ACTION PAUSE

The message indicates up or DN for each sense switch. After the sense switches have been set and all restart instructions have been followed, depress the Start key. The message ACTION COMPLETED is then printed and the restart procedure continues.

# SET KEYS TO RESTART CODE OPERATOR ACTION PAUSE

When the message is printed, the panel entry keys should be set to the restart code corresponding to the checkpoint record with which restarting is to begin (usually the last one written). The Start key should then be depressed.

### **Restart Error Messages**

# PERM. REDUN. ON CHECKPOINT MESSAGE BLOCK PRESS START TO CONTINUE

Ten consecutive atempts have been made to read the checkpoint record. When the Start key is depressed, the message condition ignored is printed and the program continues.

# INCORRECT CHECKPOINT TAPE ADDRESS CANNOT PROCEED

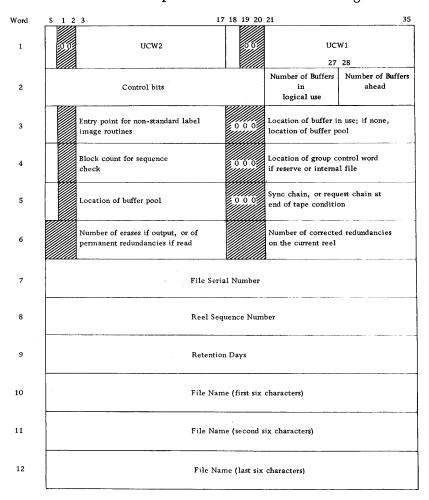
An illegal unit address for the checkpoint tape has been specified in the panel keys.

# UNIT XXXXX POSITIONING FAILED CANNOT PROCEED

The restart program has been unable to properly position tape xxxxxx.

# Appendix A: File Control Block Format

The 12 words which make up a File Control Block are arranged as follows:



The details of these twelve words are given below. Items marked with an asterisk are generated in the File Control Block by the Preprocessor. The remaining positions are initially zero.

	,	
Word(s)	Bits	Contents
1	S*	Mounting flag secondary unit (operator has been instructed to mount a reel on the secondary unit).
	1-2	Unused.
	3–17*	Unit Control Block for secondary unit. If no secondary unit is used, this is the same as bits 21—35.
	18*	Mounting flag, primary unit (operator has been instructed to mount a reel on the primary unit).
	19–20	Unused.
	21-35*	Unit Control Block for primary unit.
		Note:
		If this is an Internal File, the first word is of the
		form:
		PZE CHAIN, ,0
		where
		CHAIN = L(0) if no chain exists.

Word(s)	Bits	Contents
2	S*	Mixed mode file: 0 – No 1 – Yes
	1- 2*	Checkpoint control
		00 – No checkpoints initiated by this file.
		01 – Checkpoints are to be written on the checkpoint file at the beginning of every recof this file.
		10 - Checkpoints are to be written on this fil at the beginning of every reel of the fil (labeled output files only).
	3	0 – File is not open.
		1 – File is open.
	4	0 – Not a Reserve file.
		1 – Reserve file.
	5	0 - File is inhibited.
		1 – File is not inhibited.
	6*	Mode:
		0 – BCD
		1 – Binary
	7- 8*	File type:
		00 – Input file
		01 – Partial block output file
		10 – Total block output file
		11 – Checkpoint file
	9*	Label:
		0 – No
		1 – Yes
	10–11*	Block sequence and check sums:
		00 – No block sequence words
		10 – Block sequence word is present, no check sums.
		11 – Block sequence word is present, check sums are present (input) or are to be com puted (output).
	12*	Reel control flag: 0 – Single reel file, if unlabeled; no label search if labeled
		1 – Multi-reel file, if unlabeled; search for labe on open, if labeled
	13	Has a buffer been released:
		0 – No
		1 – Yes
	14	Has a rush occurred for this buffer:
		0 – No
		1 – Yes

Word(s)	Bits	Contents
	15	End-of-tape: 1 – No 0 – Yes
	16	File permanently closed: 0-No
	17	1 – Yes  Has a rush occurred in this sequence:  0 – No  1 – Yes
	18*	Density of label to be read: $0 - Low$ $1 - High$
	19*	Label density to be written:  0 – Low density  1 – High density
	20*	File density: 0 – Low density 1 – High density
	21–35	Count of Read buffers, or Output file chain 21–27 Number of buffers in logical use 28–35 Number of buffers ahead
3	S	Regenerative Internal file if $S = 1$
	1- 2	Unused.
	3–17	Location of entry point for standard or non-stand- ard label image routines.
	18–20	Unused.
	21–35	Location of the control word for the buffer or buffer pool to be used with this file.
4	S*	File control: 0 – Regular file 1 – Internal file
	1- 2	Unused.
	3–17	Counter for block sequence checking.
	18–20	Unused.
	21–35	Location of the Reserve Group control word, if this is a Reserve file.
5	S*	List control: $0 - \text{List this FILE card.}$ $1 - \text{Do not list this FILE card.}$
	1- 2	Unused.
	3–17	Location of the Buffer Pool.

Word(s)	Bits	Contents
5	18–20	Unused.
	21–35*	Buffer synchronization chain or location of zero cell (end of chain); Buffer request chain at end of tape condition.
6	S, 1- 2	Unused.
	3–17	Count of erase areas on the current output reel, or of permanent redundancies on the current input reel.
	18–20	Unused.
	21–35	Number of corrected redundancies for the current reel.
7	All*	File serial number in the form bXXXXX.
8	All*	Reel sequence number in the form bXXXXb.
9	All*	Retention days: bbbXXX.
10–12	All*	File name in BCD.

## Appendix B: Miscellaneous Programming Information

#### **Preservation of File Lists**

A program will normally retain a file list used to attach several files simultaneously, since the same list can be referred to later when these files are closed. In addition, any file list which contains one or more Internal or Reserve Groups *must* be preserved, at least until all such groups in the list have been closed. This is necessary because iocs reflects the current status of a group by modifying the file and buffer counts which appear in the Group control word.

## File Position After File Closing

When a file is closed, whether at the end of file position or not, it is always unbuffered, so that its physical position corresponds to the last logical usage of the file. Note, however, that in the case of an input file, information in a partially processed buffer may be lost if the file is closed and then reopened, because the tape will be positioned beyond the block which occupied this buffer.

## **Backspace File**

When the subroutine BSF is used to backspace over one or more labeled files, the structure of such files must be kept clearly in mind. For example, after an EOF exit has occurred for a labeled input file, use of BSF will produce the following results, depending on the count specified:

Count	Tape Position on Exit
1	Beyond EOF mark following the trailer label.
2	In front of the trailer label record.
3	In front of the first data block of the file.
4	In front of the first header label for the file, if it is a single reel file; or in front of the label of the current reel of a multi-reel file without checkpoints; or in front of the checkpoint record of a multi-reel file with checkpoints.
5	In front of the header label for the file, if a checkpoint was written on the file.

## Use of Copy When "Reading" with Transmitting Commands

The copy routine is normally used to transfer information located by IOXYN commands. There are certain instances, however, in which words transmitted out of a buffer will also be included in a subsequent copy to an output file. These cases are:

1. All those words in the same buffer in which one or more words were located, and which precede the last word "read," are copied. Any words beyond the last word "read" are not copied. For example, assume that the following sequence is used to "read" the first twelve words of a 20-word record:

IOCP	WS, , 3
IOCTN	** 9

If the file is now copied, the twelve words "read" will form a new 12-word record in the output file. Words 13-20 are lost from both files.

2. Even if all of the words in a buffer are transmitted to working storage, the buffer is still connected to the file unless the last word was "read" with an IOCD OF IORY command. The transmitted words can then be copied. For example, if the block is "read" by

IOCT

WS, , BLOCK-SIZE

it may be copied. If, however, the block is "read" by

**IORT** 

WS, , \*\*

it cannot be copied, because the IORT command has disconnected the buffer from the file.

#### Re-use of a Buffer Pool

When all of the files attached to a Buffer Pool have been closed, the structure of the pool is essentially the same as when it was first defined, and a new set of files may be attached to it. Note that any Reserve or Internal Groups must also have been closed by an appropriate reference in a close list.

## **Joining Pools**

When one Buffer Pool is joined to another, it must have the exact form of a pool which has just been defined. Because any use of a pool will, in general, alter the order in which its buffers are chained, such a previously used pool must be redefined before it is joined to another.

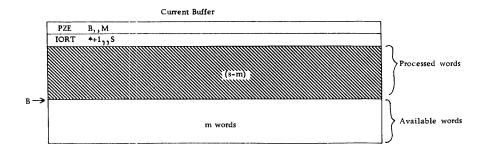
## **Actions of IOCS Routines Under Abnormal Conditions**

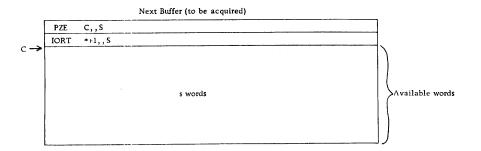
Routine	Condition	Action
DEFINE	Attempt to define a buffer pool inside the system boundaries	Stop: Pool Error
JOIN	POOL2 in use (or already joined to a different pool)	Stop: Pool Error
	POOL1 joined to any buffer pool	Stop: Pool Error
ATTACH	Attempt to attach a closed file which has been rewound and unloaded	Stop: Initialization Error
	Attempt to attach to pool in use	Not tested
	Not enough buffers in a buffer pool	Stop: Attach Error
CLOSE	Close a closed file	NOP, except that rewind option is used
OPEN	Attempt to open a file which was re- wound and unloaded	Stop: Initialization Error

Routine	Condition	Action
READ	File never opened	EOF exit
	File already closed	eof exit
	Block sequence error	error exit $MQ_{s-2} = 1$
	Check sum error	ERROR exit $MQ_{s-2} = 2$
	Redundancy error	error exit $MQ_{s-2} = 4$
	Sequence and redundancy errors	error exit $MQ_{8-2} = 5$
	Check sum and redundancy errors	error exit $MQ_{8-2} = 6$
	Attempt to locate information in two buffers with one command	еов exit $MQ_{8-2}=2$
	Attempt to locate in more buffers than are available	EOB exit $MQ_{8-2} = 4$
	Attempt to transmit a word into the system	Stop: Illegal Transmit
WRITE	File never opened	Normal exit
	File already closed	Normal exit
	Attempt to locate information in two buffers with one command	еов exit $MQ_{S-2}=2$
	Attempt to locate in more buffers than are available	EOB exit $MQ_{S-2} = 4$
COPY	Either file not open	NOP
	No buffer(s) connected to the input file	NOP
	Files not on same pool	NOT TESTED
REW	File never opened	NOP
	File already closed	NOP
	File is labeled	NOP
WEF	File never opened	NOP
	File already closed	NOP
	File is input file	NOP
	File is labeled	NOP
BSF	File never opened	NOP
	File already closed	NOP
	BSF one file after EOF exit	NOP
BSR	File never opened	NOP
	File already closed	NOP
	Following EOF exit	вог exit
	Tape positioned in front of the first rec- ord of a file	вог exit
STASH	Either file not open	NOP
	Case 1: No buffer in FILE1 chain	nts exit
	Case 2: FILE2 group full	NTS exit
	Case 4: FILE2 group full	NTS exit

## **IOCS Command Execution Tables**

The details of the execution of the iocs commands are presented in the two tables which follow. Each case described is presented in reference to the following diagram:





#### where

B-the location in the current buffer of the first available word

m-the number of available words in the current buffer

C-the location of the first word in the next buffer

s-the size of all buffers in the pool

In the tables which follow, the general form of the rocs command executed is:

#### where

A—the location in working storage to be "read" into or "written" from N—the number of words to be processed

Indirect addressing of commands is not shown, however, it may be specified for all 10cs commands except TCH.

In the two tables there are several actions indicated for buffer release or truncation, these are:

Hold—the end of the buffer was passed, however, it will not be released to the pool (READ) or written (WRITE) until next reference to the file, because it contains words located by this READ or WRITE sequence.

Conditional Hold—the buffer will go into HOLD status if some IOXYN command were executed on it during the current READ or WRITE sequence.

#### **Execution of IOCS Commands Read**

EOB = 0

EOB + 0

_	OCS mmand
IOCY	$A,,n$ $n \le m$ $n > m$
IOCD	$A,,n$ $n \le m$ $n > m$
IOSY	$A,,n$ $n \le m$ $n > m$
IORY	A,,**

EOD-0		
History Record in AC	Words Read	Buffer Re- leased
PZE A + n,, m - n PZE A + n,, s - n + m	n n	no 1st
PZE A + n,, m - n PZE A + n,, s - n + m	n n	yes both
PZE A + n,, m - n PZE A + m,, 0	n m	no no yes
	History Record in AC  PZE A + n , , m - n PZE A + n , , s - n + m  PZE A + n , , s - n + m  PZE A + n , , s - n + m  PZE A + n , , s - n + m	History Record in AC Words Read  PZE A + n,, m - n PZE A + n,, s - n + m  PZE A + n,, s - n + m  PZE A + n,, s - n + m  PZE A + n,, s - n + m  PZE A + n,, m - n PZE A + n,, m - n PZE A + m,, 0

	_		
History Record	Words	Buffer Release	EOB
in AC	Read	Action	exit
PZE A + n,, m - n	n	no	no
PZE A + n,, s - n + m	n	Conditional Hold 1st	no
PZE A + n,, m - n	n	Conditional Hold	no
PZE A + n,, s - n + m	n	Conditional Hold both	no
PZE A + n,, m - n PZE A + m,, 0 PZE A + m., 0	n m	no no Conditional Hold	no no
	mAC  PZE A + n,, m - n  PZE A + n,, s - n + m  PZE A + n,, s - n + m  PZE A + n,, s - n + m  PZE A + n,, s - n + m	In AC   Read   PZE A + n,, m - n   n   n	in AC  PZE A + n,, m - n PZE A + n,, s - n + m  PZE A + n,, s - n + m  PZE A + n,, s - n + m  PZE A + n,, m - n PZE A + m,, m - n PZE A +

CVII

CVI	n
OVI	Ų

IOCYN	**,,n
	$n \leq m$
	n > m
IOCDN	**,,n
	$n \le m$
	n > m
IOSYN	**,,n
	$n \leq m$
	n > m
IORYN	**,,**

IOCYN B,,n	PZE B + n,, $m - n$	n	no
IOCYN C,,n	$\frac{\text{PZE C} + \text{n} - \text{m}_{,,s} - \text{n} + \text{m}}{\text{PZE C} + \text{n} - \text{m}_{,s} + \text{m}}$	n	1st
IOCDN B,,n	PZEB+n, m-n	n	yes
IOCDN C,, n	PZEC+n-m,,s-n+m	n	both
IOSYN B,,n	PZEB+n,,m-n	n	no
IOSYN B,, n	PZE B + m,,0	m	no
IORYN B,, m	PZE B + m,, 0	m	yes

	PZE B + n,, m - n	n	no	no
	PZE B + m,, 0	m	Hold	code 2
	PZE B + n,, m - n	n	Hold	no
	PZE B + m,, 0	m	Hold	code 2
IOSYN B,, n IOSYN B,, n IORYN B,, m	PZE B + n,, m - n PZE B + m,, 0 PZE B + m,, 0	n m m	no no Hold	no no

## Execution of IOCS Commands Write

EOB = 0

EOB	+	0
-----	---	---

	OCS nmand
IOCY	$\begin{array}{c} A,, n \\ n \leq m \\ n > m \end{array}$
IOCD	$\begin{array}{c} A,, n \\ n \leq m \\ n > m \end{array}$
IOSYN	$\begin{array}{c} A,, n \\ n \leq m \\ n > m \end{array}$
IORY	$A,,n \\ n \leq m \\ n > m$

		<del></del>	
IOCS Command changed to	History Record in AC		Buffer tten
	$ \begin{array}{c c} PZE A + n, m - n \\ PZE A + n, s - m + n \end{array} $	n n	no 1st
	$ \begin{array}{c} PZE A + n, m - n \\ PZE A + n, s - m + n \end{array} $	n n	yes all
	PZE A + n,, m - n PZE A + n,, s - n	n n	no 1st
<u>-</u>	PZE A + n,, m - n PZE A + m,, 0	n m	yes yes

IOCS Command changed to	History Record in AC	Words "writ- ten"	Buffer Release _ Action	EOB exit
	$ \begin{array}{c c} PZE A + n, m - n \\ PZE A + n, s - m + n \end{array} $	n n	no Conditional Hold 1st	no no
	$ \begin{array}{c} PZE A + n, m - n \\ PZE A + n, s - m + n \end{array} $	n n	Conditional Hold Conditional Hold both	no no
	PZE A + n, m - n $PZE A + n, s - n$	n n	no Conditional Hold 1st	no no
	PZE A + n,, m - n PZE A + m,, 0	n m	Conditional Hold Conditional Hold	no no

SKIP

## LOCATE

IOCYN	$     \begin{array}{l}       **, , n \\       n \leq m \\       n > m     \end{array} $
IOCDN	**,,n n≤m n>m
IOSYN	**,,n n≤m n>m
IORYN	**,,n n≤m n>m

IOCYN B,, n	PZE B + n,, m - n	n	no
IOCYN C,, n	PZE C + n - m,, s - m + n	n	1st
IOCDN B,, n	PZE B + n,, m - n	n	yes
IOCDN C,, n	PZE C + n - m,, s - m + n	n	all
IOSYN B,, n	PZE B + n,, m	0	no
IOSYN C,, n	PZE C + n,, s		1st
IORYN B,,n	PZE B + n,, m - n	n	yes
IORYN B,,n	PZE B + m,, 0	m	yes

IOCYN B,, n	PZE B + n,, m - n	n	no	no
	PZE B + m,, 0	m	Hold	code 2
IOCDN B,,n	PZE B + n,, m - n	n	Hold	no
IOCDN B,,n	PZE B + m,, 0	m	Hold	code 2
IOSYN B,, n	PZE B + n,, m	0	no	no
IOSYN C,, n	PZE C + n,, s		Conditional Hold 1st	no
IORYN B,, n	PZE B + n,, m - n	n	Hold	no
IORYN B,, n	PZE B + m,, 0	m	Hold	no

## Glossary

The following glossary gives definitions of terms as used in this manual. The definitions do not necessarily apply to other systems.

ATTACH The rocs routine which establishes the connection between a group of files and the buffer pool which they use. Every file, except a checkpoint file, must be attached to some buffer pool.

AVAILABLE BUFFER A buffer not presently in use nor reserved for later use.

AVAILABLE BUFFER COUNT The number of currently available buffers in a pool, kept by rocs for each buffer pool.

AVAILABLE WORD LOCATOR The first word of each buffer, to keep track of the location of the next available word and the number of available words remaining in the buffer.

AVAILABLE WORDS Those words in an input buffer which have not yet been "read," or, in an output buffer,

which have not yet been filled with data.

An intersystem supervisor which facilitates the continuous operation of many individual systems. See the bulletin entitled: IBM 7090 Operating Systems: Basic Monitor (IBSYS). BASIC MONITOR

**BLOCK** A physical record; that is, a tape record, a card, or a line of print.

**BLOCKING** The arrangement of data into blocks whose size is convenient and efficient for processing.

BLOCK COUNT The number of blocks processed on the current reel of a file, kept in the corresponding File

Control Block.

**BLOCK CHECK SUM** An 18-bit folded check sum of the data words in a block, which appears in positions S-17 of

the Block Sequence Word.

**BLOCK SEQUENCE NUMBER** The position of a data block among the other data blocks in the current reel of a file. This

number is found in positions 21-35 of the Block Sequence Word.

**BLOCK SEQUENCE WORD** A word which can be appended to each block of a binary file in order to provide additional

checking information.

**BSF** The iocs routine which backspaces a tape to the beginning of the first data block of some file

on that reel.

BSR The IOCs routine which backspaces one data block of a file. The routine will not backspace

across an end-of-file mark.

BUFFER An area assigned for rocs to use as an intermediate storage area for data which are to be trans-

mitted between storage and the input/output devices.

**BUFFER CONTROL** The control used by a class of 10cs commands which treat all, or the remainder, of a buffer

as a data group.

A group of equal sized buffers connected in such a way as to permit sharing of buffers among a group of files. BUFFER POOL

**BUFFER SIZE** The maximum number of data words which a buffer can hold. This restricts output blocks to

that maximum length and determines the maximum number of words which can be read from

an input block.

CHECKPOINT A reference point at which tape positions, the status of machine registers and switches, and the

contents of storage are recorded for the purpose of later restarting the program.

CHECKPOINT FILE A tape specified to receive checkpoints which are initiated by the operator, or the program, or by reel switching on other files. The checkpoint file should not be attached, but must be

opened. It cannot be labeled.

CLOSE The rocs routine used to terminate file usage, prepare end of file trailer labels, and rewind the

file.

CKPT The locs routine which prepares a checkpoint on the current checkpoint file. COMMAND (IOCS)

Specified in the calling sequence to the READ or WRITE routines, it controls the "reading" or "writing" of data. The general form of an iocs command is ioxy(N)(\*).

CONTROL CARDS

A group of cards which are used by the Preprocessor to initialize a job run. The six types of control cards are job, file, date, restart, load, and ibsys.

CONNECTED BUFFER

A buffer currently in use by a file.

COPY

The rocs routine which transfers data from an input file to an output file by connecting processed input buffers to that output file.

COUNT CONTROL

The control used by a class of iocs commands which "reads" or "writes" a specified number of words

CREATION DATE

The date a file was prepared. It is contained in the label of a labeled file.

DATE

A Preprocessor control card which introduces the current date to be used in the writing and checking of labels.

**DEFINE** 

The rocs routine used to impose the structure of a buffer pool upon a specified area of storage.

DISCONNECT (buffer)

To release a processed buffer from a file.

EOB (end of buffer)

A programmed switch, specified in the calling sequence to the READ and WRITE routines, which determines the system interpretation of non-transmitting rocs commands. Under certain circumstances it is used as an exit from those routines.

EOF (end of file)

An address, specified in the calling sequence to the READ routine, which is used as an exit upon recognition of an end of file condition.

ERR (error exit)

An address, specified in the calling sequence of the READ routine, which is used as an exit upon recognition of an uncorrectable redundancy, or a block sequence or check sum error.

EOT (end of tape)

An address, specified in the calling sequence of the WEF routine, which is used as an exit upon recognition of the end-of-tape condition.

FILE

A body of data, recorded on some external medium, which can be processed by the iocs routines.

FILE BLOCK

The area of storage which contains all the File Control Blocks for a program.

FILE BLOCK ORIGIN

The location of the first word of the File Block, specified as an octal number on the JOB card.

FILE CARD

A Preprocessor control card which describes a file to be processed by 10cs.

FILE CONTROL BLOCK

12 cells allocated for control information about a file 10cs is to process. The File Control Block is prepared by the Preprocessor from the symbolic information on a file card.

FILE LIST

A list of the files to be attached to a buffer pool.

FILE SERIAL NUMBER

The Tape Serial Number of the first reel of a labeled file. This number must always be specified for a labeled input file.

HEADER LABEL

A record at the beginning of a reel of tape which identifies the reel.

**IBSYS** 

The symbolic name for the Basic Monitor.

IMMEDIATE FILE

A file which is opened automatically by the ATTACH routine. Immediate Files cannot be reopened by the user after being closed.

INTERNAL FILE

A file which is contained entirely within a chain of buffers in core storage. Such a file has no external I/O unit associated with it.

INTERNAL GROUP A group of Internal files, preceded in a file list by an Internal Group control word. INTERNAL GROUP A word with prefix six which precedes a group of Internal files in a file list. It specifies the CONTROL WORD buffer requirements of the group, and the number of files in the group which can be in use simultaneously. I/O UNIT Allowable units in Iocs are any tape, card reader, punch, or printer attached to a data channel. JOB CARD A Preprocessor control card which identifies a job and specifies the File Block Origin and length of the File Block. **JOIN** The IOCS routine which connects two separately defined buffer pools, forming a larger pool whose buffers are not necessarily contiguous. LABEL A record at the beginning (header label) or end (trailer label) of a reel of tape, which contains control information for that reel. LABELED FILE A tape file, each reel of which begins with a header label and ends with a trailer label. LOCATE In IOCs, the process of determining the location of data in a buffer. Locating is accomplished by the use of IOXYN commands. LOGICAL CHANNEL A designation for the group of Unit Control Blocks for all 1/0 units assigned to a single data LOGICAL RECORD Data which, as a group, has some logical significance to the user. MULTI-REEL FILE A tape file which is contained in more than one reel. rocs can process labeled multi-reel files and unlabeled multi-reel files which contain only one EOF mark on each reel. MWRThe IOEX routine which prints messages on the system printer. NON-TRANSMITTING Those IOCS commands, of the form IOXYN, which provide the location of data within a buffer, COMMANDS rather than transmitting the data between buffers and working storage.

**OPEN** 

The IOCS routine which does the initialization necessary in order to "read" or "write" a file.

**PREPROCESSOR** 

The part of rocs which creates the File Control Blocks required by a program and acts as the locs monitor.

READ

The IOCS routine which, by direction of IOCS commands, "reads" input data from a file.

RESERVE FILE

A file which belongs to some Reserve Group. It may be opened and closed any number of times during the course of a program.

RESERVE GROUP

A group of files (preceded in a File List by a Reserve Group control word) whose exact buffer requirement can be specified by the user.

RESERVE GROUP CONTROL WORD A word which precedes each group of Reserve Files in a File List. It specifies that group's buffer requirement and the number of files in the group which can be in use simultaneously.

RESTART

The restoration of the contents of storage, the status of switches and registers, and the position of tapes to the condition held at the time a checkpoint was taken, and continuing the program from there.

RETENTION DAYS

The number of days a file is to be saved after its creation.

REW

The rocs routine which can be used to rewind an unlabeled file.

SKIP

In 10cs, the process of bypassing data words in a buffer while "reading" or "writing." Skipping is accomplished by use of the 10xxn commands.

SPECIAL COUNT CONTROL

The control used by a class of rocs commands which process by count unless terminated by the end of buffer condition. These commands are of the form rosy(n).

STASH

The rocs routine which transfers data from an Internal file to a regular file or vice versa, or between Internal files, by reconnecting buffers from one file to another.

TAPE SERIAL NUMBER

A number appearing in a tape label which identifies that reel of tape. It is usually the same as the external identification of the reel.

TRAILER LABEL

A record at the end of each reel of a labeled file which indicates whether or not that reel is the last reel of the file.

TRANSMIT

In iocs, the process by which ioxy commands move data between buffers and the user's working storage.

TRUNCATE

To ignore any remaining words in a processed buffer and release it from a file; for an input file, the buffer is made available; for an output file, action is initiated to write the contents of the buffer on that file's 1/0 unit.

UNBLOCKING

The separation of blocked data into its data groups.

UNBUFFERING

The process of adjusting the physical position of a tape to correspond to its logical position. For an output file, all buffers in use are truncated, and a delay occurs until all writing ceases. For an input file, all buffers in use are returned to the pool, and the tape is positioned to a point following the block corresponding to the buffer which contained the last word "read."

UNIT CONTROL BLOCK

A group of four control words in IOEx of Basic Monitor for each I/O unit on the machine.

WEF

The rocs routine which can be used to write an EOF mark on an unlabeled output file.

WRITE

The rocs routine which, by direction of rocs commands, "writes" output data on a file.

# Index

additional header records       21, 26         ATTACH       6, 7, 19, 24, 70, 75         attachment       1, 6, 7, 27         available       75         buffer       5, 14, 75         words       14, 75	configuration options       42         control card format       42         control sense switches       55         control words       11         Buffer       14         COPY       2, 18, 42, 69, 70, 71, 76         count control commands       10, 11, 13         creation date       22, 24, 42, 76
file       20, 69         record       20         block       3, 75         count       23, 24, 75         sequence error       17, 39         blocking       75         BSF       20, 42, 69, 70, 75         BSR       20, 42, 71, 75         buffer       4, 75         availability       19, 27         commands       11, 14	data file       3         data transmission       2, 10, 18         DATE card       42, 45, 76         deferred opening       27         DEFINE       4, 70, 76         density       40         conventions       21, 35         disconnect buffer       76         dummy label       24
control word         11           definition         1           requirements         6           input files         6           output files         6           partial block         6           total block         6           Immediate files         6           Internal files         7           Reserve files         7           truncation         11, 14, 16, 19           buffering         1, 26           Buffer Pool         1, 4, 19, 70, 75           control word         4           joining         70	effective address computation       11         end of available buffers       29         end of file trailer       23         end of reel trailer       23         EOB       13, 15, 16, 28, 29, 31, 39, 76         EOF       15, 16, 24, 26, 31, 39, 76         EOF label       23, 24         EOR label       23         EOT       19, 76         ERR       15, 16, 39, 76         Error History Control Sense Switch       55, 59         examples       transmitting commands       12         non-transmitting commands       14         Executor       42, 43
length       4         re-use of       70         buffers       1         calling sequences       15         Chain Address       5         Checkpoint Control Sense Switch       55         checkpoint       34, 39, 42, 44, 66, 75         identification       63         record       21, 22, 35         check sum       22, 33, 44, 66         error       16, 17, 39         CKPT       34, 42, 75         CLOSE       18, 19, 31, 70, 75         rewind options       19         closing files       2, 18, 19, 31         command list       12         termination       16         command execution table       72         completed program return       40         conditional hold       72	false redundancy       17         file       3, 76         attachment       1, 7, 27         block       46, 76         block origin       39, 42, 43, 76         closing       2         density       22, 44         description       1         initialization       1         list       7, 57, 58, 82         preservation of       69         name       23, 44         number       43, 44         opening       1, 6, 9, 23, 26         position       69         processing       2, 10         processing messages       61, 62         serial number       22, 23, 44, 76         specifications       1         FILE card       23, 35, 42, 43, 76

File Control Block 1, 3, 23, 27, 33, 38, 40, 41, 65, 76	blank tapes
reserving space for	files
folded check sum	tape format
halts	labeling
system 58	routines
header labels	messages
history record	labels
hold	LOAD Card 42, 45
	locate 2, 77
IBSYS Card 42, 46	locating 13, 16, 69
ID Card	logical
IOEX 36, 38	channels
Immediate files 6, 8, 76	record
buffer requirements 7	look-ahead bits
closing	
indirect addressing	messages
initialization 1	file processing
input buffers Immediate files 6	label 59, 60, 61
Immediate files6Internal files7	Preprocessor
Reserve files	restart
Input Control Sense Switch	tape mounting
Input files 6, 44, 66	multi-file reels
Internal files	multi-reel unlabeled files
attachment 7	MWR
closing 31	33, 1
opening 30	, 1 1
Internal Group 30, 31, 32, 69, 77	nonstandard
control word	labeling routines       25         labels       24
IOCD 11,70	nontransmitting commands
IOCDN	examples
IOCP	NTS 18, 31, 32, 71
IOCPN       13         IOCS commands       2, 10, 11, 75	, , ,
(see also specific command)	on-line messages
IOCS operation	OPEN 9, 19, 30, 70, 77
IOCS routines	opening files
IOCT 11	operator checkpoint
IOCTN	output
IORP 11	buffers 6
IORPN 14	files 44, 66
IORT 11	type-P
IORTN 14	type-T 6, 44, 66
IOSP	overlapping
IOST	
IOSTN 14	parameters
2921	physical block
[OB card	physical files
job name	physical record
JOIN 4, 42, 70, 77	Pool
	control word 4
label 77	length 4
checking 23	sample 5
error	Preprocessor
information	error messages 56, 57
Label Control Sense Switch	primary input/output unit
labeled	program

input       46         input unit       38, 46, 55         loading       1, 3, 46         preparation       38         programming using IOCS       3	special count control       10, 12, 77         special routines       36         standard label formats       21         start procedure       56         STASH       18, 31, 42, 71, 78         suppression of error history       55
READ       2, 10, 13, 15, 16, 17, 24, 26, 28, 31, 71, 77         "read"       2, 10, 11, 12, 13, 17, 18, 69, 70         redundancy       16, 17, 55         error       62         reel switching       23, 24, 26         Reserve files       6, 7, 9, 19, 77	system       39         halts       58         operation       56         tape       38, 56
Reserve files       6, 7, 9, 19, 77         attachment       27         buffer requirements       7         closing       18, 19         opening       9         Reserve Group       7, 13, 27, 28, 69, 77         control word       8, 19, 69, 77         reserving space for File Control Blocks       3         RESTART Card       42, 45         restart       34, 77         code       63         messages       63, 64         Restart Control Sense Switch       55, 63         retention days       22, 24, 77         REW       20, 42, 71, 77         rewinding       24         labeled files       20         rewind options       9, 19	tape       59         removal messages       59         serial number       22, 78         TCH       12         terminating command list       16         trailer labels       21, 23, 24, 78         end of file       23         end of reel       23         transfer of information       17         transfer points       38         transmitsion       2, 10, 12, 18, 78         transmitting commands       11, 16         examples       13         truncate       11, 14, 16, 19, 78         type-P output files       6, 44         type-T output files       6, 44         unbuffering       20, 78
sample       5         Buffer Pool       5         problem       47         secondary input/output unit       43, 65         sense printer       37         sense switch assignments       55         sequence       16, 17         numbering       22, 23, 33, 44, 66         single reel unlabeled files       25         skipping       13, 16, 77	Unit Control Blocks       41, 65, 78         files       26         file procedures       26         using a file for input and output       27         using more than one buffer for a file       28         variable length records       53         WEF       19, 42, 71, 78         WRITE       2, 10, 13, 16, 17, 31, 71, 78         "write"       2, 10, 11, 12, 17

# **Publications**

Following is a list of IBM publications which may be of interest to the reader:

## REFERENCE MANUALS

Form Number	Title
A22-6528-1	IBM 7090 Data Processing System
A22-6536	IBM 709 Data Processing System
A22-6616	7340 Hypertape Drive

## **GENERAL INFORMATION MANUALS**

Form Number	Title
D22-6508-2	IBM 709/7090 Data Processing System
F28-8001	Sorting Methods for IBM Data Processing Systems
F28-8043	IBM Commercial Translator
F28-8053-1	COBOL
F28-8074-1	FORTRAN

## **BULLETINS**

Form Number	Title
G22-6505-1	IBM 7090 Data Processing System
G22-6634	7340 HYPERTAPE Drive
J28-6080	IBM 709 Utility Programs
J28-6098-1	FORTRAN Assembly Program (FAP) for the IBM 7090
J28-6132	Advance Specifications: 7090 FORTRAN and FORTRAN Assembly Program (FAP)
J28-6138	Sort 709: Sorting Times for the IBM 7090 with IBM 729 VI Magnetic Tape Units
J28-6152	IBM 7090 with IBM 7340 HYPERTAPE Drives: Programs and Programming Systems
J28-6153	IBM HYPERTAPE Input/Output Control System for 7000 Series Data Processing Systems
J28-6156	IBM 7090 Generalized Sorting Program Using IBM 7340 HYPERTAPE Drives
J28-6166	SHARE 7090 9PAC: Part 1—Introduction and General Principles
J28-6167	SHARE 7090 9PAC: Part 2—The File Processor
J28-6168	SHARE 7090 9PAC: Part 3—The Reports Generator
J28-6169	IBM 709/7090 Commercial Translator Processor
J28-6173	IBM 7000/1400 Output Editing System—Preliminary Reference Manual
J28-6174	S-Program for the IBM 7090: Preliminary Specifications
J28-8086	7090 Operating Systems: Basic Monitor (IBSYS)