File Number 7040-30
Form C28-6309-3

IBM Systems Reference Library

IBM 7040/7044 Operating System (16/32K) Input/Output Control System

This publication describes the IBM 7040/7044 Input/Output Control System for 16/32K IBM 7040/7044 Data Processing Systems — 7040-IO-952. The Input/Output Control System is a set of routines which perform the input/output operations required by both the Operating System and application programs.

Among the subjects discussed in this publication are the relationship between the Input/Output Control System and the 7040/7044 Operating System, the use of the three programming levels of the Input/Output Control System, the capabilities for teleprocessing, and the specifications of the 1401 Input/Output Control Program for use with an on-line 1401.

Preface

This publication describes the IBM 7040/7044 Input/Output Control System, hereafter referred to as IOCS, and the methods by which the programmer can use the system to increase the efficiency of his programs and decrease the time necessary for writing and testing them. Because IOCS can be used in three distinct ways, the publication consists of a general introduction, one chapter on each of the three main levels of IOCS, and a chapter on the label handling facilities of the system. It is suggested that the user read the introduction and the chapter describing the section of IOCS that most interests him.

The use of this IOCS requires a 7040/7044 Data Processing System that has an extended performance instruction set, 16, 384 or more words of core storage, and at least the minimum of input/output units required by the 7040/7044 Operating System.

The reader must be familiar with the IBM 7040/7044 Data Processing Systems, the principles of coding for symbolic assembly programs, and the use of the Operating System. To satisfy these requirements, the reader should be familiar with the contents of the following publications:

IBM 7040/7044 Systems Summary, Form A28-6289

IBM 7040/7044 Principles of Operation, Form A22-6649

IBM 7040/7044 Operating System (16/32): Programmers' Guide, Form C28-6318

IBM 7040/7044 Operating System (16/32K): Macro Assembly Program (MAP) Language, Form C28-6335

MAJOR REVISION (July 1964)
This edition, Form C28-6309-3, supersedes the previous edition, Form C28-6309-2, and associated Technical Newsletters, Forms N28-0500 and N28-0506. It includes a discussion of the teleprocessing capabilities added to the Input/Output Control System for the 1414 Input/Output Synchronizer, Models IV and V. Changes are indicated by a vertical line placed to the left of the affected text.

Copies of this and other IBM publications can be obtained through IBM Branch Offices.
Address comments concerning the contents of this publication to:
IBM Corporation, Programming Systems Publications, Dept. D39, 1271 Avenue of the Americas, New York, N. Y. 10020

Contents

Introduction

The purpose of an input/output control system is to reduce the amount of time spent by the programmer in writing and testing the input/output portions of his programs. To perform this function, IOCS acts as an intermediary between the object program and the input/output devices attached to the computer, and allows installation and system control over input/output units and operations.

Most input/output routines require coding effort disproportionate to the part played by input/output in the logic of the program. They also constitute a major source of errors and often require numerous debugging runs. The IOCS removes both difficulties by providing the programmer with a simplified programming environment that closely fits the logical input/output requirements of his program and by providing pretested routines to substitute for coding that he would ordinarily have to supply.

Basic Concepts

The following sections form a general description of the input/output characteristics of the IBM 7040/7044 Data Processing Systems and contain the definitions of many terms that occur throughout this publication. To the experienced programmer, many of the explanations will be elementary, but some of the terms being defined have restricted uses for the purposes of this publication. For this reason, programmers who are thoroughly familiar with the subjects under discussion may find it useful to read the sections that follow for the definitions of terms.

File

One of the basic concepts of data processing is the *file.* A file is a collection of related data. It may be recorded on cards, magnetic tape, disk storage, or some other recording medium. It is important to maintain a distinction between the file and the recording medium on which it resides.

A file is recorded as a series of *blocks,* or *physical records,* in the recording medium. Each block may contain one or more data records.

The *data record* is the basic unit of information for a data processing program. It consists of a single datum, or piece of information, from a file. A data record may be a single number, it may be all of the information pertaining to a given business transaction, or it may be a record of the value of several parameters at a given point in an experiment. The logical structure of most data processing programs consists of reading, processing, and writing individual data records. For this reason, data records are sometimes termed *logical records.*

Because of the characteristics of the various input/output units on which a file may be held, the problems of transforming data from physical records on one unit, to data records in core storage, and back to physical records on another unit, can become fairly complicated. One of the advantages of using IOCS is the ability to read and write physical records on the various input/output units almost interchangeably. It is for this reason that the concepts of "file" and "data record" must be independent of the device used.

Throughout this publication, the term *end of data file* will be used for the actual end of the data file. The "end of file" on a card reader, for instance, indicates that all of the cards in the hopper have been read; but the file to which these cards belong may be continued on a magnetic tape unit or on some other device.

The storage capacities of such devices as magnetic tape units and disk storage are finite, as contrasted with card punches, which can write unlimited amounts of data, provided that the hopper is kept filled with blank cards. When no more data can be written on a magnetic tape or disk storage unit, the *end of medium* is said to have been reached. If data remains to be written in the output file that is being placed on this *primary unit,* the writing must take place on an *alternate unit.* To accomplish this, the primary and alternate units are *switched.* IOCS greatly simplifies the process of unit switching.

The only alternative to unit switching on 729/7330 magnetic tape units is changing tapes on the same unit. This process is extremely costly in terms of time, since the computer is usually idle while the operator is changing the tape.

Labels

A *label* is a record that identifies a file, or part of a file. Labels come at the beginning and end of each segment of the file. The first label of each segment is the *header* label, and the second is the *trailer* label.

Labels are used to distinguish among the various segments of a file. For example, if a file is held on five reels of magnetic tape, the header labels indicate the order in which the reels should be read, and the trailer label on each reel indicates whether or not the file has come to an end. Each reel before the last has an *end-of-reel trailer label.* The last reel ends in an *end-of-file trailer label.*

Labels also contain such information as the recording mode of the file, the length of time that the file is to be retained, and the date on which the file was created.

Data Channels and Overlap

The devices that control the transmission of data from the central processing unit to the various input/output units are the data channels. All 7040 and 7044 Data Processing Systems are equipped with at least one data channel (channel A). Up to four more (channels B-E) may be added as optional features.

Central processing unit operation is suspended while information is being transmitted between core storage and an on-line 1401 or a 1414 Input/Output Synchronizer on channel A. This is called *nonoverlapped* operation. Channels B through E, however, allow central processing unit operation to occur simultaneously with input/output operation on one or more of them. This *overlap* of processing and input/output allows the main program to read ahead on a file, that is, to read data records into core storage while previously read data records are being processed. It also allows the program to process data records while previously processed data records are being written out from core storage.

The user of IOCS need not concern himself with the differences between channel A and channels B-E, except when he is programming at the IOEX level of IOCS, described in the chapter "Input/Output Executor (IOEX)."

Buffers

Each physical record must be read into or written out from a set of consecutive locations in core storage. These input/output areas are called *buffers.* Each buffer contains exactly one physical record, although the record may not always fill the buffer completely.

It is necessary to have more than one buffer for each file being read or written, if overlapping is to be used. One buffer holds the information that is in the process of being read or written and the other(s) holds the data being processed.

The data (i.e., logical) records in each block of an input file must be separated and presented to the processing sections of the program one at a time. This process is known as *deblocking.*

A buffer is refilled after all of the data records have been removed from it. As soon as processing of all the data records in a given buffer has been completed, the buffers reverse roles, from read-in buffer to *current buffer* (i.e., the buffer from which data records are being extracted). This is *buffer switching.*

One or more output data records, created by the processing section of the program, are placed in a buffer *(blocking).* When a given buffer can hold no more data records, its contents are written out and buffers are switched, so that another buffer is ready to accept further data records while the preceding records are being written out from the first buffer.

The IOCS provides tested routines to block and unblock data records, to switch buffers as each one becomes empty or full, and to initiate the reading and writing necessary to maintain a continuous flow of information.

It is not usually necessary for the programmer to reserve his own buffers, provided that he is using the relocatable Loader and the buffer processing routines of IOCS. The relocatable Loader (IBLDR) has provisions for creating the buffers necessary for IOCS use when the object program is being loaded into core storage. (The publication *IBM 7040/7044 Operating System (16/32K): Programmer's Guide,* Form C28-6318, contains detailed information on this process.) This is normally done from the specifications of each file, given by the programmer.

Structure of IOCS

The 16/32K Input/Output Control System is a collection of routines, located in the lower part of core storage. It is divided into five sections.

The Input/Output Buffering System (IOBS) contains the routines that perform the blocking, deblocking, and buffer switching operations of IOCS. The features of IOBS largely relieve the programmer of the necessity of considering the characteristics of the medium on which each of his files is recorded. They allow him to concentrate on the processing steps in his program, by providing or accepting individual data records.

Input/Output Operations (IOOP) performs all of the reading, writing, and non-data-transmitting operations required by IOCS. The IOOP level consists of two sections: IOOP1 contains the calling sequence interpretation routine, the unit synchronizer routine, and the select and error-recovery routines for 729/7330 magnetic tape units and 1301 Disk Storage or 7320 Drum Storage; IOOP2 contains the select and error-recovery routines for the 1009 Data Transmission Unit, the 1011 Paper Tape Reader, the 1014 Remote Inquiry Unit, the 1402 Card Read Punch, the 1403 Printer, the 1622 Card Read Punch, and telegraph units.

The Input/Output Label System (IOLS) performs unit switching, verifies and creates IBM standard labels, and uses IOOP to read and write labels.

The Input/Output Executor (IOEX) attends to data channels traps, schedules the use of the data channels for input/output operations, schedules the use of the central processing unit for programs of various priorities, and services central processing unit traps.

The order in which the sections of IOCS reside in core storage is shown in Figure 1.

If the storage protection optional feature is available to the system, it is used to protect at least the

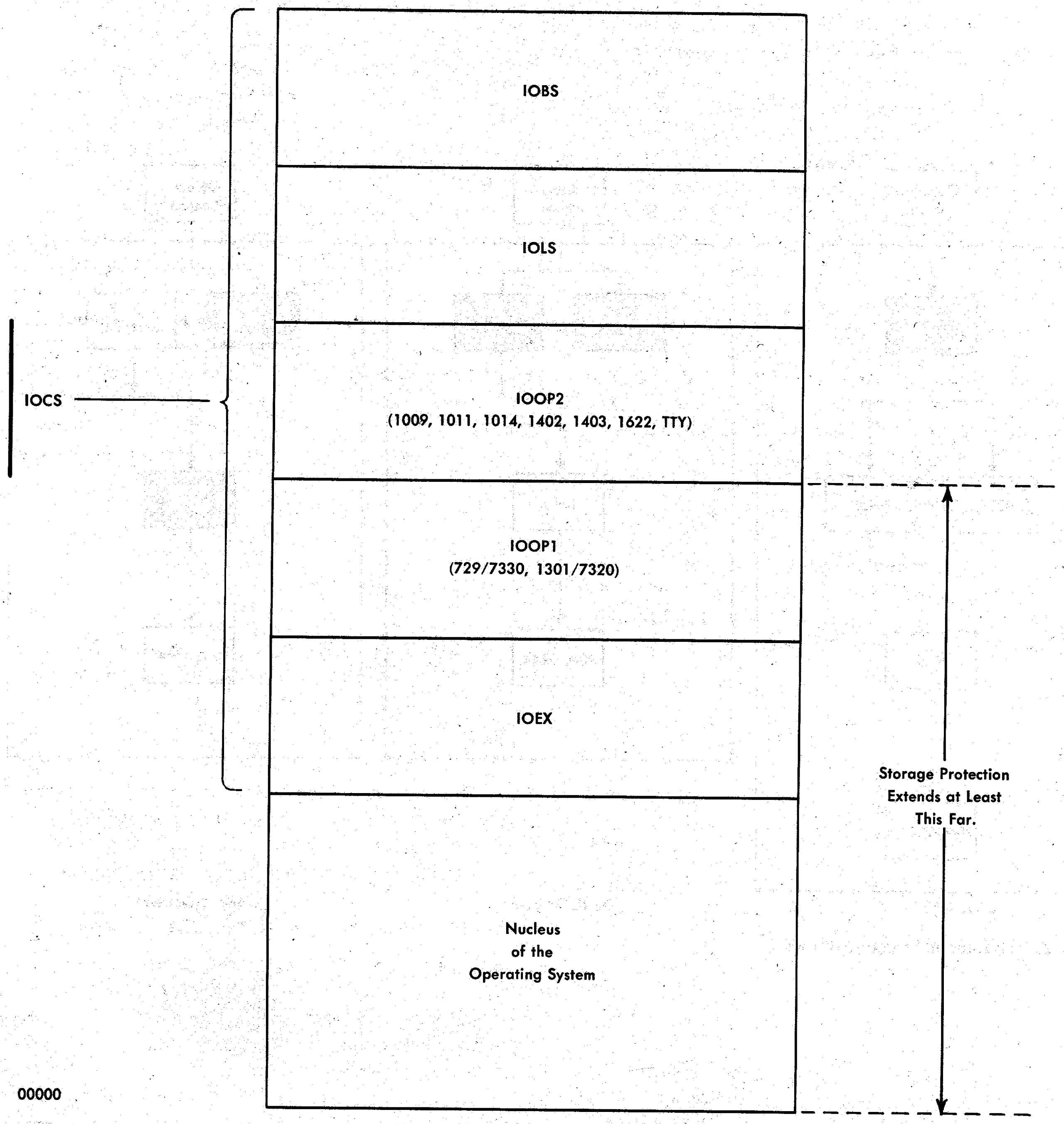


Figure 1. Arrangement of the Parts of the Input/Output Control System

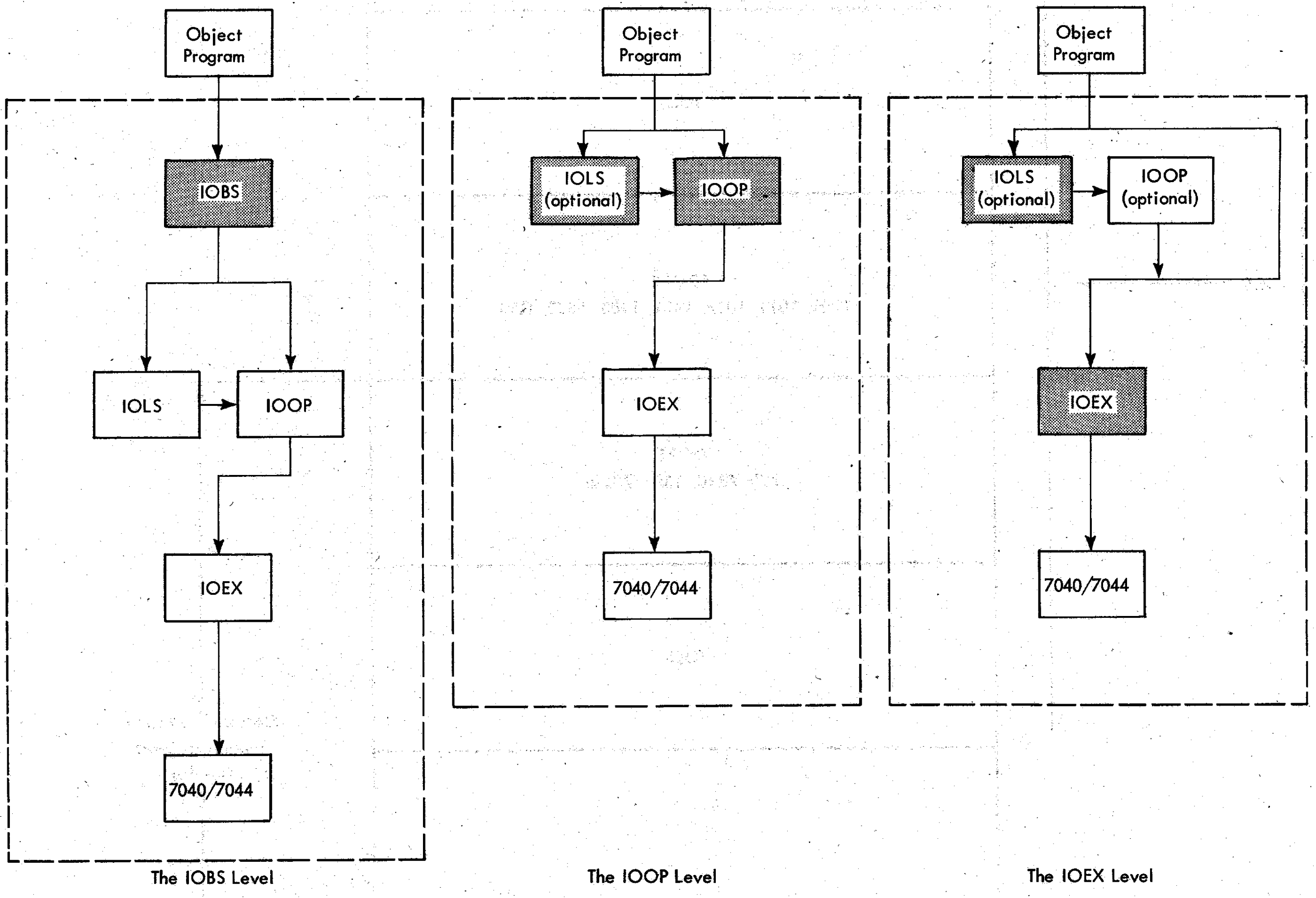
portions of IOCS used by the System Loader (S.SLDR): IOEX and IOOP1.

Levels of IOCS

The structure of IOCS permits three distinct approaches to input/output programming. The sections of IOCS that form the three levels of IOCS are shown in Figure 2. The shaded blocks indicate the section with which the object program communicates.

The programmer should choose one level of input/output programming for each file being processed and should use that level exclusively for all input/output operations required for that file. The use of two or more levels of IOCS on a given unit may result in errors.

In general, it is recommended that the highest level of IOCS consistent with the size and input/output requirements of the program be used. However, there are certain restrictions placed on the choice of IOCS levels to be used by the programmer. A no-priority object program may call any IOCS routine; a special routine may not call IOBS or IOLS; a non-standard select and error-recovery routine may not call IOBS, IOLS, or IOOP.



The IOBS Level

The IOOP Level

The IOEX Level

Figure 2. The Internal Structure of IOCS

IOCS and the Operating System

The programmer can specify to the Loader (IBLDR) the level of IOCS that is used by his object program. The loader uses this information to relocate the origin of the program to a point above the end of the highest section of IOCS being used, while taking into account the fact that the origin of the program cannot be within a protected part of core storage.

To increase the flexibility of the Data Processing System and to speed job-to-job transition, the input/output units that are read and written by IOCS are designated symbolically, rather than by actual addresses. The symbolic units are listed in Figure 3. The correspondence between an actual input/output device and a symbolic unit is specified by a series of tables in the Nucleus of the Operating System. The operator can vary this correspondence by using System Monitor control cards or the entry keys.

The number of utility units at an installation is determined at system assembly.

Symbol	Unit
S.SLB1	System Library Unit 1
S.SLB2	System Library Unit 2
S.SOU1	System Output Unit 1
S.SOU2	System Output Unit 2
S.SIN1	System Input Unit 1
S.SIN2	System Input Unit 2
S.SPP1	System Peripheral Punch Unit 1
S.SPP2	System Peripheral Punch Unit 2
S.SCK1	System Checkpoint Unit
S.SU00	Utility Unit 0 (available to the object program)
S.SU01	Utility Unit 1 (available to the object program)
.	.
.	.
.	.
S.SU99	Utility Unit 99 (available to the object program)

Figure 3. Table of Symbolic Units

Input/Output Buffering System (IOBS)

The Input/Output Buffering System (IOBS) allows the programmer to use a logical approach to his file processing requirements. After the physical arrangement of each file has been specified, the programmer simply requests input or output of data records. The IOBS blocks and deblocks data records, switches buffers, and reads and writes physical records, as required.

Use of IOBS

The first step in the use of IOBS is file description. Files may be described in the programmer's coding through the FILE and LABEL pseudo-operations of the Macro Assembly Program language. If the files are unlabeled, only the FILE pseudo-operation need be used. Each labeled file, however, requires a LABEL pseudo-operation immediately following the FILE description.

The programmer may choose to describe his files at load time, rather than at assembly time. He may also wish to change certain options that were specified by FILE pseudo-operations. In either case, he may use the Loader (IBLDR) control cards $FILE and $LABEL. The specifications in these cards override certain preassembled specifications, and $FILE and $LABEL cards for files that are not as yet specified perform the same function as FILE and LABEL entries in the MAP coding. The FILE and LABEL pseudo-operations are described in the publication *IBM 7040/7044 Operating System (16/32K): Macro Assembly Program (MAP) Language,* Form C28-6335. The $FILE and $LABEL control cards, and the $POOL control card (discussed later), are described in the publication *IBM 7040/7044 Operating System: Programmer's Guide,* Form C28-6318.

At load time, when the object program is being relocated by the Loader, the label and file-description information about each file is translated into a file control block. The location of the first word of the file control block replaces the name of the file in the user's coding. The file control block contains both permanent information, such as labeling data, and temporary information, such as the location of the first word of the next data record to be processed.

Also at load time, the specifications of the various files are inspected to determine the size and number of buffers required. Ordinarily, one buffer of appropriate size is created for each single-buffered file, and two buffers are created for each double-buffered file. If the user wishes, however, he may cause fewer than the usual number of buffers to be created for a group of files that have approximately the same buffer size requirements. This is done by the use of the $POOL Loader control card. The $POOL card specifies the size of the individual buffers, the total number of buffers to be created, and the files that will use these buffers. As each file is opened, it is assigned one or two buffers from the pool, depending on whether it is single- or double-buffered. When it is closed, the buffers are returned to the pool. The number of buffers in the pool need only be equal to the maximum number of buffers required by the files in the pool that are open simultaneously. If some of the files are closed before others are opened, some buffers may be used by two or more files, considerably reducing the space required.

The format of a buffer pool is:

```
pool        PZE        list,,m
list        PZE        area1+1,,length
            PZE        area2+1,,length
             .          .
             .          .
             .          .
            PZE        aream+1,,length
area1       BSS        length+2
area2       BSS        length+2
 .           .          .
 .           .          .
 .           .          .
aream       BSS        length+2
```

where:

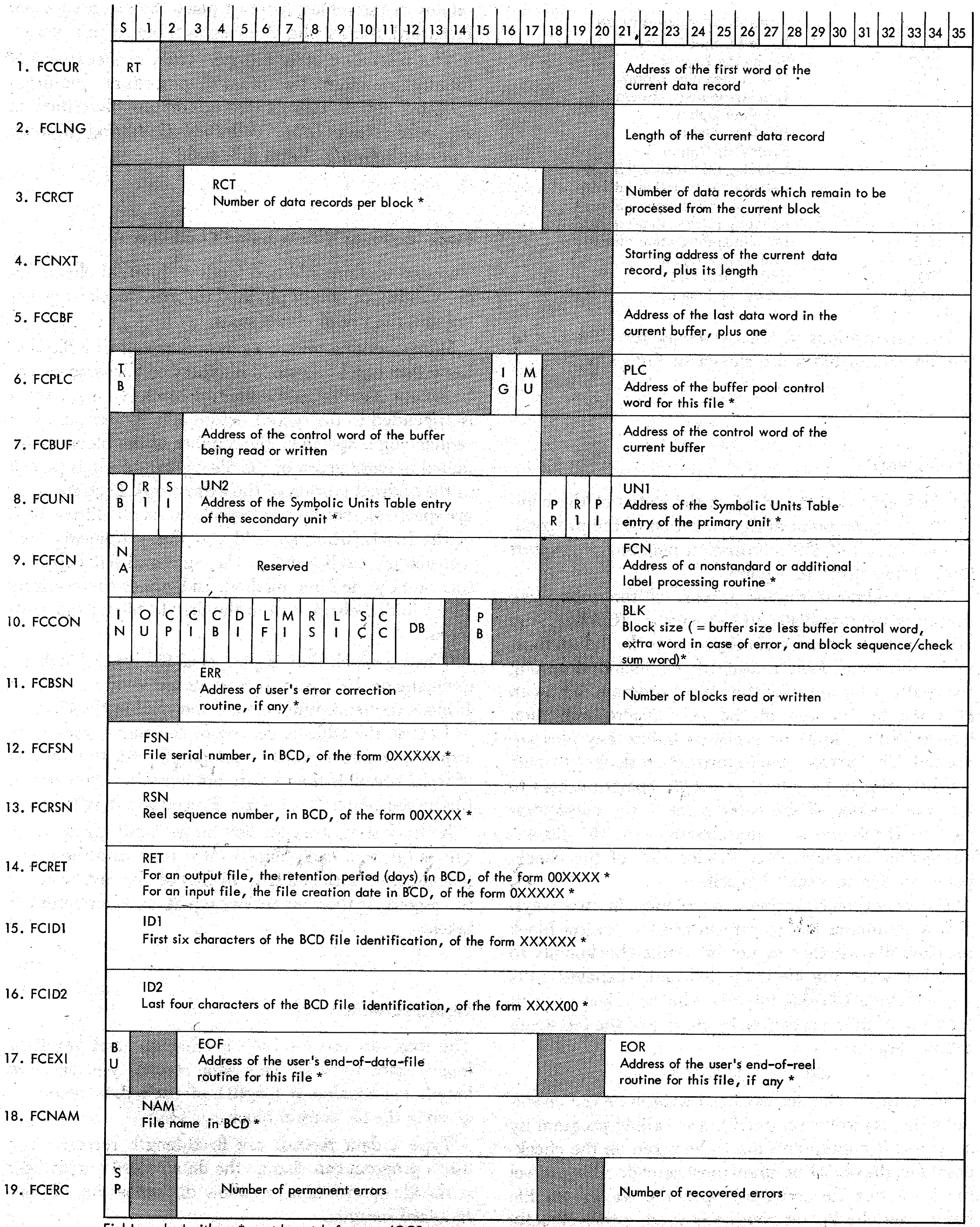
m
is the number of buffers in the pool.

length
is the length of each buffer, including Type 2 and 3 data record control words and the check sum/block sequence number word (if any).

Only the elements of the list need be in consecutive core storage locations.

File Control Blocks

Each file control block contains 19 words of information concerning a specific file. This information is used, during the processing of the file, both by the main program and by IOBS. The format of the file control block is shown in Figure 4.



	S	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31	32	33	34	35
1. FCCUR	RT		▨																			Address of the first word of the current data record														
2. FCLNG	▨																					Length of the current data record														
3. FCRCT	▨			RCT Number of data records per block *															▨			Number of data records which remain to be processed from the current block														
4. FCNXT	▨																					Starting address of the current data record, plus its length														
5. FCCBF	▨																					Address of the last data word in the current buffer, plus one														
6. FCPLC	TB	▨															IG	MU	▨			PLC Address of the buffer pool control word for this file *														
7. FCBUF	▨			Address of the control word of the buffer being read or written															▨			Address of the control word of the current buffer														
8. FCUNI	OB	RI	SI	UN2 Address of the Symbolic Units Table entry of the secondary unit *															PR	RI	PI	UN1 Address of the Symbolic Units Table entry of the primary unit *														
9. FCFCN	NA	▨		Reserved															▨			FCN Address of a nonstandard or additional label processing routine *														
10. FCCON	IN	OU	CP	CI	CB	DI	LF	MI	RS	LI	SC	CC	DB		▨	PB	▨					BLK Block size (= buffer size less buffer control word, extra word in case of error, and block sequence/check sum word)*														
11. FCBSN	▨			ERR Address of user's error correction routine, if any *															▨			Number of blocks read or written														
12. FCFSN	FSN File serial number, in BCD, of the form 0XXXXX *																																			
13. FCRSN	RSN Reel sequence number, in BCD, of the form 00XXXX *																																			
14. FCRET	RET For an output file, the retention period (days) in BCD, of the form 00XXXX * For an input file, the file creation date in BCD, of the form 0XXXXX *																																			
15. FCID1	ID1 First six characters of the BCD file identification, of the form XXXXXX *																																			
16. FCID2	ID2 Last four characters of the BCD file identification, of the form XXXX00 *																																			
17. FCEXI	BU	▨		EOF Address of the user's end-of-data-file routine for this file *															▨			EOR Address of the user's end-of-reel routine for this file, if any *														
18. FCNAM	NAM File name in BCD *																																			
19. FCERC	SP	▨		Number of permanent errors															▨			Number of recovered errors														

Fields marked with an * must be set before any IOCS routine is used.

Figure 4. Format of the File Control Block

The fields in the file control block are set as follows:

FIELD	SET BY FILE DESCRIPTION
RCT	Record Count Option
PLC	Set by the Loader
UN1	Unit Assignment Options
UN2	Unit Assignment Options
FCN	Labeling Option
BLK	Blocksize Option
ERR	Error Exit Option
FSN	Labeling information (if any)
RSN	Labeling information (if any)
RET	Labeling information (if any)
ID1	Labeling information (if any)
ID2	Labeling information (if any)
EOF	End-of-File Option
EOR	End-of-Reel Option
NAM	Set by the Loader

The explanations of the two-character bit codes in the file control block are shown in Figure 5.

Checkpoints

A checkpoint is a recording of the status of the computer at a given point in the execution of the program. It contains all of the information necessary to restart the program from that point.

The checkpoint routine is part of the subroutine library of the IBM 7040/7044 Processor (16/32K). Any program that requires checkpoints to be taken must cause the checkpoint routine to be included among the routines loaded with the object program. This can be done by the use of the MAP macro-instruction CHKPNT, which must be executed before any files are opened. The CHKPNT macro-instruction does not cause a checkpoint to be taken. It simply communicates to IOCS the location of the entry point to the checkpoint routine. If the user does not choose to use this macro-instruction, he must place the location of the checkpoint routine in S.SCKT himself.

The checkpoint routine can be used in two ways. The programmer can specify in the file control block for each file whether or not he wants checkpoints to be taken when the file is opened and whenever units are switched. Checkpoints can also be taken at arbitrary points in the program, by the use of the following calling sequence:

```
            TSX         S.CKPT,4
```

After the checkpoint has been written on the checkpoint file, control is returned to the calling program by a TRA 1,4. If checkpoints are to be taken on the checkpoint file, the object program must provide a file control block for this file and must open the checkpoint file before the checkpoint routine is used, either directly or indirectly. If the user is assembling his own file control blocks, rather than using FILE macro-instructions or $FILE control cards, and checkpoints are to be taken on any of these files, he must place the location of the first word of his string of file control blocks in S.SFBL.

The restarting of a program from a checkpoint is usually an externally initiated procedure, requiring operator action. Restart procedures are described in the publication, *IBM 7040/7044 Operating System: Operator's Guide*, Form C28-6338.

Block Sequence Numbers and Check Sums

The IOBS can provide two kinds of internal checks on the validity of binary physical records: block sequencing and the use of check sums.

Whenever IOBS writes a physical record of a file that has either block sequence numbers or check sums (or both) specified in its file control block, an extra word is appended to the record before it is written. If block sequencing is specified, the number of the block, maintained in word FCBSN of the file control block, is placed in the address portion of the extra word. If check sums are specified, IOCS takes a logical sum of all of the words of the block [using an Add and Carry Logical (ACL) instruction], adds together the right and left halves of this sum by the same method, and places the resulting 18-bit folded check sum in the first 18 bits of the extra word.

When a block that is part of a file using block sequencing or check sums is read, its last word is checked. If block sequence numbers are specified in the file control block, the address portion of the check sum/block sequence number word is compared with the address of word FCBSN. If the two are not equal, an error return (discussed under "IOBS Error Procedures") is taken.

If check sums are specified for an input file, a check sum is taken of each block as it is read, and the result is compared with the first 18 bits of the last word of the record. If the two are not equal, an error return is taken.

Record Formats

The IOBS can process both fixed-length and variable-length data records. The data record type and the length (or maximum length) of each data record is given in the file control block.

Type 1 data records are fixed-length records. The user's program can change the data record length field in the file control block to allow the processing of nonstandard formats.

Type 2 and Type 3 data records are standard format variable-length data records, in which the length and

Bit Code	Description	Explanation	Set by
RT	Record Type*	= 00 for Type 1 records. = 10 for Type 2 records. = 11 for Type 3 records.	Record Type Option
TB	Truncation Bit	= 0 if the current buffer is not to be truncated on the next entry to the IOBS for this file. = 1 if the current buffer is to be truncated on the next entry.	IOBS
IG	Ignore Get	= 0 if no error exit was taken for error type 4 or 10. = 1 if error exit was taken for error type 4 or 10.	IOBS
MU	Mixed Unit Record	= 0 if file is not defined as unit record Type 3. = 1 if file is defined as unit record Type 3.	IOBS
OB	Open Status Bit	= 0 if the file is not open. = 1 if the file is open.	IOBS
R1, R2	These bits are reserved for future use by DS Programming Systems.		
SI	Secondary Unit Deferred Mounting Message*	= 0 if a mounting message for the secondary unit is not to be typed when a reel switch occurs. = 1 if a mounting message for the secondary unit is to be typed when a reel switch occurs.	Mounting Option
PR	Permissible to Reposition	= 0 permissible to reposition at CLOSE. = 1 do not reposition at CLOSE.	IOLS
PI	Primary Unit Deferred Mounting Message*	= 0 if a mounting message for the primary unit is not to be typed when the file is opened. = 1 if a mounting message for the primary unit is to be typed when the file is opened.	Mounting Option
NA	Nonstandard or Additional Label Processing	= 0 if the label is a standard IBM label and is to be verified by IOCS. If the address in bits 21-35 of this word is not zero, this address is the location of the entry point to an additional processing routine. = 1 if the label is not a standard IBM label and is not to be verified by IOCS. The label is read into and written from the label area. If the address in bits 21-35 of this word is not zero, this address is the location of the entry point to a nonstandard verification or creation routine.	Labeling Option
IN	Input	= 0 if the file is not input. = 1 if the file is input.	IOBS
OU	Output	= 0 if the file is not output. = 1 if the file is output.	IOBS
CI	Checkpoint Indicator*	= 0 if there are no checkpoints on this file. = 1 on an input file, if a checkpoint follows the header label on each reel of the file, and is to be automatically passed over. = 1 on an output file, if a checkpoint is to be written following every header label.	Checkpoint Options
CP	Checkpoint*	= 0 if no checkpoints are to be taken on the checkpoint file for this file. = 1 if a checkpoint is to be taken on the checkpoint file when this file is opened and whenever units are switched on this file.	Checkpoint Options
CB	Checkpoint File*	= 0 if this is not the checkpoint file. = 1 if this is the checkpoint file.	Checkpoint File Option
DI	Density Indicator**	= 0 if the file is in low density. = 1 if the file is in high density.	File Density Option
LF	List File**	= 0 if the file is to be listed at the end of the job. = 1 if the file is not to be listed.	File Disposition
MI	Mode Indicator	= 0 if the file is in BCD mode. = 1 if the file is in binary mode.	IOBS
RS	Reel Specification*	= 0 if the file is single-reel. = 1 if the file is multi-reel. (Unlabeled files only)	Reel Handling Option
LI	Labels Indicator*	= 0 if the file is unlabeled. = 1 if the file is labeled.	Presence of Label Card
SC	Sequence Check*	= 0 if no block sequence numbers are to be formed or checked for this file. = 1 if block sequence numbers are to be formed or checked for this file.	Block Sequence Option
CC	Check Sum Check*	= 0 if no check sums are to be formed or checked for this file. = 1 if a check sum is to be formed or checked for each record written or read on the file.	Check Sum Option
DB	Disposal Bits**	= 00 if the file can be destroyed at the end of the job. = 01 if the file is to be printed at the end of the job. = 10 if the file is to be punched at the end of the job. = 11 if the file is to be held at the end of the job.	File Disposition
PB	Prime Buffers	= 0 do not prime buffers. = 1 prime buffers at first call to IOBS for this file.	IOBS
BU	Buffers*	= 0 if the file is single-buffered. = 1 if the file is double-buffered.	Buffer Option
SP	Special Condition	= 0 if no special conditions exist. = 1 if file is defined as unit record Type 3 and/or error exit was taken for error type 4 or 10.	IOBS

Fields marked with an * must be set before any IOCS routine is used; Bits marked with a ** are not interpreted by IOBS. They are used by some other part of the Operating System.

Figure 5. Explanation of File Control Block Bits

recording mode (BCD/binary) of the record are specified in the record itself.

The first word of each Type 2 or Type 3 data record specifies the kind and amount of information in the record. The first word of a binary Type 2 or Type 3 data record is of the form:

101	n	010		x
S 2	3 17	18 20	21 29	30 35

where:

n
is the number of words in the data record, excluding the control word itself.
x
is one of the control characters listed in Figure 6.

The prefix and the tag in this word are ignored by IOBS. They are included for compatibility with other systems.

The first word of a BCD Type 2 or Type 3 data record is of the form:

n	n	n	n	n	x
S 5	6 11	12 17	18 23	24 29	30 35

where:

nnnnn
is a BCD number (right-justified) that specifies the number of characters in the data record, including the six characters in the control word itself.
x
is one of the control characters listed in Figure 6.

The control character for each data record, *x*, should be one of the characters shown in Figure 6.

These codes allow *mediary* records (that is, records that may be required for further processing later in the program or may be used as input to another program at a later date) to be intermixed with records that are to be listed or punched.

In processing Type 2 input records, IOBS ignores control characters. When Type 2 records are being written, IOBS inserts a control character K into the control word of BCD data records and a control character M into the control word of binary data records.

Mixed mode files being read from or written on unit record devices should be considered to consist of Type 3 records. The physical record is not blocked, does not contain a control word, and must not exceed the maximum physical record length of the device.

Although the length of each Type 2 or Type 3 data record is specified in the control word of the record, the size of a single data record is limited by the size of a single buffer, as data records can never be continued from one buffer to another.

Control Character	Indicates that This Record Is:	Indicates that The Next Record Is:
2	a BCD record to be printed.	BCD.
3	a BCD record to be printed.	binary.
4	a BCD record to be punched.	BCD.
5	a BCD record to be punched.	binary.
6	a binary record to be punched.	binary.
7	a binary record to be punched.	BCD.
K	a BCD mediary record.	BCD.
L	a BCD mediary record.	binary.
M	a binary mediary record.	binary.
N	a binary mediary record.	BCD.
O	a BCD record to be ignored.	BCD.
P	a BCD record to be ignored.	binary.
Q	a binary record to be ignored.	binary.
R	a binary record to be ignored.	BCD.

Figure 6. Type 2 and 3 Data Record Control Characters

Opening

Before processing of a file can begin, the file must be opened. That is, the file control block must be initialized, buffers assigned to the file, and the primary input/output unit positioned to the first record of the file.

The format of the calling sequence used for opening a file is:

```
        TSX        S.OPEN,4
        pfx        file,,d
```

where:

pfx,
S bit = 0 if the unit is to be rewound before processing begins.
= 1 if the unit is not to be rewound before processing begins.
1 bit = 0 if the file being opened should be processed in BCD mode.
= 1 if the file being opened should be processed in binary mode.
2 bit = 0 if the file being opened is an input file.
= 1 if the file being opened is an output file.

file
is the location of the first word of the file control block for the file being opened (i.e., the name of the file).

d
is ignored if the prefix, *pfx*, specifies that the primary unit is to be rewound. If the unit is not to be rewound and the file control block specifies that the file is labeled, then *d* has the following significance:
d = 0; do not check the header label on the primary unit.
d = 1; check the header label on the primary unit.

When this calling sequence is given, IOBS initializes the file control block by setting the mode and input/output bits to agree with the calling sequence and setting the cumulative fields to zero. If there are not enough buffers available to fill the buffering requirements of the file, the job is terminated.

If the file is labeled and either the calling sequence specifies a rewind or the decrement of word 1 of the calling sequence is 1, the header label of the file is either read into core storage and verified or created and written out, depending on whether the file is an input or output file.

If the file control block specifies that a checkpoint follows the header label on each reel of the file, this checkpoint is taken if the file is output or is skipped over if the file is input. If the file control block specifies that a checkpoint be placed in the checkpoint file whenever a new reel is begun on the file currently being opened, such a checkpoint is taken if there is an open checkpoint file.

If the user wishes to open several files at a time, he can use the following calling sequence:

```
        TSX        S.OPNL,4
        PZE        a,t,n
```

where:

a,t

is the location of the first of a series of words like word 1 of the S. OPEN calling sequence, each defining one of the files to be opened and the actions to be taken for that file.

n

is the number of files to be opened.

If one or more files require checkpoints to be taken on the checkpoint file, and that file is already open, a single checkpoint is taken after all of the files specified by the list have been opened.

Input File Processing

When the programmer wants to be provided with a single data record (as specified in the file control block) from a file, he uses the following calling sequence:

```
        TSX        S.GETL,4
        PZE        file,,tmtadd
```

where:

file

is the address of the first word of the file control block of the file from which the record is to be obtained (i.e., is the name of the file, as specified in the FILE pseudo-operation or the $FILE card).

tmtadd

is the location of the area into which the data record is to be transmitted by IOBS.

If the address of the transmit area, *tmtadd*, is left blank, the data is located in the buffer, but is not transmitted.

Processing Type 1 Input Records

When the S.GETL calling sequence is given for an input file consisting of Type 1 records, IOBS compares the number of data records contained in each block (if this is specified in the file control block) with the number of data records already provided from the current buffer. If it is possible that the current buffer contains another data record, the data record length field in the file control block is compared with the number of words left in the buffer. If the number of words left in the buffer exceeds the record size, the address of the first unused data word in the buffer is placed in the address portion of word FCCUR. If *tmtadd* is nonzero, the data record is transmitted to the area that begins with *tmtadd*.

If the number of data records that have been supplied from the current buffer is equal to the maximum number of data records in a block, or if there are not enough words remaining in the buffer to satisfy the record length requirements in the file control block, buffer switching occurs and any data words remaining in the buffer are lost. The IOBS then attempts to find the required number of data words in the next buffer. If there are not sufficient words in the new block to satisfy the requirements for a data record, the error routine specified by the file control block is entered (as described in "IOBS Error Procedures" at the end of this chapter). The job is terminated if no error routine has been indicated.

Processing Type 2 Input Records

When requested to GET a Type 2 data record, IOBS assumes that the next data word of the file is the control word for the next data record. It uses this word (performing BCD to binary conversion, if necessary) to determine the number of words in the record and places this information in the address portion of word FCLNG in the file control block. The location of the word immediately following the control word is placed in word FCCUR. If the length of the record exceeds the number of words remaining in the buffer, the error return is taken (the job is terminated if no error return is available). If a change of mode is indicated, the error return is taken. If a transmission area has been specified by the calling sequence, the data in the record is moved to this area, beginning at location *tmtadd.*

Processing Type 3 Input Records

The major difference between the method of processing used by IOBS for Type 2 data records and that used for type 3 data records is in the handling of control characters.

When control is returned to the main program after IOBS has obtained a Type 3 data record, the six low-order bits of the accumulator contain a version of the control character that was found in the control word of the record. The control character that is left is one of the following characters: 2, 4, 6, K, M, O, or Q. The change-of-mode indication is considered internal to the operations of IOBS and is not transmitted to the user.

Processing Input Files by Blocks

If the user wants to obtain an entire block (i.e., physical record) of data or the remainder of the current block of data, he can use the following calling sequence:

```
          TSX       S.GETB,4
          pfx       file,,tmtadd
```

where *file* and *tmtadd* are defined in the same way that they were for S.GETL, and:

pfx
= PZE if the user wants the next complete block of the file.
= MZE if the user wants the remainder of the current block of the file.

If *pfx* is PZE, IOBS begins reading a new block into the current buffer, places the location of the first data word of the next buffer in the address portion of FCCUR, and places the number of words that were read into that buffer in the address portion of word FCLNG.

If *pfx* is MZE, the address of the first unused data word in the current buffer is placed in FCCUR, and the number of data words remaining in the buffer is placed in FCLNG. If there are no more data words in the current buffer, the address portions of both FCCUR and FCLNG are set to zero.

If *tmtadd* is nonzero, the data located by this operation is moved from the buffer to the transmission area that begins with location *tmtadd.*

Non-Data-Transmitting Operations on Input Units

The IOBS allows several non-data-transmitting operations to be used with input files.

Backspace: To backspace over one block, the programmer can use the following calling sequence:

```
          TSX       S.BSR,4
          PZE       file,,bof
```

The symbol *file* is the location of the first word of the file control block, and *bof* is the location of a routine to be entered if the unit is already at the beginning of the file, where no backspacing can take place.

When this operation is performed, the file is unbuffered, the unit assigned to the file is backspaced, and the block count, in word FCBSN of the file control block, is reduced by one. If the block count is already zero, the *bof* return is taken. The result of a backspace operation on a mixed mode file is unpredictable.

Force End-of-Reel: If the programmer wishes to simulate the end-of-reel condition on a file, he can use the following calling sequence:

```
          TSX       S.FEOR,4
          PZE       file
```

When this calling sequence is used, IOBS types any deferred mounting message, rewinds and unloads the unit from which records were being read, rewinds the alternate unit for the file, switches units, and, if the file is labeled, checks the header label on the new unit. If the file control block indicates that a checkpoint appears on the file, this checkpoint and its file mark are passed over. If the file control block indicates that a checkpoint should be written on the checkpoint file when units are switched, this checkpoint is written.

Rewind: If the programmer wishes to rewind an unlabeled file, he can use the following calling sequence:

```
          TSX       S.REW,4
          PZE       file
```

If this calling sequence is given for a labeled file or an unopened file, the job is terminated.

Label Handling

The IOBS uses the Input/Output Label System (IOLS) to perform all label verification and creation procedures. If an input file is described as labeled by its file control block, the following label operations may occur during the processing of the file.

Header Label: When the file is opened, and each time that units are switched, the new header label is verified to ensure that the correct reel or unit is being used. If the label is in error, it is typed, along with an operator message, and a pause occurs. The operator may cause the reel to be accepted, mount a new reel (to be verified in turn), or terminate the job.

Trailer Label: When a file mark is read, the end-of-reel return is taken. When the end-of-reel routine returns control to IOBS, the trailer label is read and checked. If it indicates the end of the file, the end-of-file exit specified in the file control block is taken. If the trailer label indicates the end of the reel, the old unit is rewound and unloaded, unit switching occurs, the new unit is rewound, and the header label on the new unit is verified. If the trailer label is in error, it is typed, along with an operator message, and a pause for operator action occurs.

Output File Processing

When a programmer wishes to place a single data record in a file, he has several ways to proceed. If he wants to have the record transmitted from a work area into the output buffer, or to locate space in an output buffer, he can use the following calling sequence:

```
TSX     S.PUTL,4
PZE     file,,tmtadd
```

where:

file
is the location of the first word of the file control block.

tmtadd
is the location of the first word of the area from which data is to be transmitted.

IOBS locates, in an output buffer, the number of words specified in the data record length field of the file control block. If *tmtadd* is nonzero, data is transmitted from the area beginning with *tmtadd* into the space that has been located. If *tmtadd* is zero, IOBS performs no transmission, and it is assumed that the located words have been filled by the main program when the next reference is made to the file.

If, on the other hand, the programmer does not know the number of words to be placed in the output buffer, he can use the following calling sequence to locate space in an output buffer:

```
TSX     S.PLOC,4
PZE     file,,n
```

where:

n is the maximum number of words to be used.

The IOBS locates space for *n* data words in the current buffer, if possible, and places the location of the first word available for data in FCCUR. If space for *n* data words is not available in the current buffer, this buffer is truncated and written out, and the location of the first available word of the alternate buffer is placed in FCCUR. If there are not *n* available words in the alternate buffer (presumably empty), the error return for the file is taken. If the file control block specifies Type 2 or Type 3 data records, IOBS reserves one word, immediately preceding the *n* words located for data, to serve as a control word.

When the programmer has completed his processing and transmission of data into the buffer, he updates the file control block by using the following calling sequence:

```
TSX     S.PUTL,4
MZE     file,,m
```

The MZE prefix indicates that the data is already in the buffer, and *m* is the number of words that were actually used of the *n* that were located. Since the buffer displacement field of the file control block is not altered until this calling sequence is given, the number *m* may be less than *n*, without any loss of buffer space. If another record is placed in the same buffer, it begins immediately after the *m*th word of the previous record. The logical record length specification in the file control block is set to the value of *m*, when the MZE prefix is used.

Processing Output Data Records

In summary, if the user wishes to place records in the output buffer, he can use one of the following approaches:

1. Type 1 fixed-length data records can be placed in an output file by a call to S.PUTL, with or without transmission.

2. If transmission is desired, the user can specify the length of the record by a call to S.PLOC, which locates the specified number of words. The actual number of words to be transmitted is given by the logical record length specification in the file control block. If desired, the user can change this specification.

3. If transmission is not desired, the process used is the one described in the previous section; i.e., using a call to S.PLOC to locate *n* words and then using a call to S.PUTL (with the prefix of the second word an MZE) to specify the number of words actually filled. Note that the logical record length is updated to *m*.

4. The user can place variable-length data records in an output buffer by adjusting the data record length field in the file control block himself and then calling S.PUTL with or without transmission.

When a call is made to S.PLOC, the address portion of FCCUR is set to the address of the word immediately following what will be the control word for the record. The control word for a Type 2 or Type 3 record is not actually constructed, however, until a call to S.PUTL is made. The length of the record is then calculated, converted to BCD if necessary, and placed in the control word.

If the file control block for a file of Type 2 records specifies that the file is BCD, a control character K is placed in bits 30-35 of the control word. If the file is binary, the control character is M.

When the user makes a call to S.PUTL for a Type 3 record (or to S.PLOC if he is locating space), he must leave a control character that specifies the mode of the current record (2, 4, 6, K, M, O, and Q) in the six low-order bits of the accumulator. If this character indicates a mode different from that of the last data record, the control character of the first data record in the current buffer is changed to indicate that the following block is to be read in the opposite mode, the current buffer is truncated and written out, and space is located for the user's request in the next buffer.

Processing Output Files by Blocks

If the user wants to PUT a complete block of information into the file, he can use the following calling sequence:

```
TSX    S.PUTB,4
PZE    file,,tmtadd
```

When this calling sequence is executed, IOBS writes out all of the data in the current buffer, sets the buffer displacement field in FCNXT to zero, and places the starting address and length of the next buffer in the address portions of FCCUR and FCLNG, respectively. If the file is of Type 2 or Type 3 records, no control word is created.

The next time any IOBS routine is called for this file, the buffer is truncated and written out. If the buffer displacement field is still set to the address of the first word of the buffer, the entire buffer is written. Otherwise, only the data between the beginning of the buffer and the address specified by the buffer displacement field is written.

If *tmtadd* is nonzero, a group of words equal in number to the specified blocksize of the file is placed in the buffer. Not all of these words are necessarily written, however, as the buffer displacement field may still be changed to exclude some of them.

If the user wishes to PUT a variable size block of information into the file, he may use the following calling sequence:

```
TSX    S.PUTB,4
MZE    file,,m
```

When this calling sequence is executed, IOBS truncates the current buffer and writes out the first *m* words of data in the buffer. The starting address of the next buffer is placed in the address portions of FCCUR and FCNXT. The blocksize and logical record length specifications in the file control block remain unchanged. If the file is of Type 2 or Type 3 records, no control word is created. The buffer to be PUT using S. PUTB with an MZE prefix, can be filled either by placing data directly into it, or by calling S. PUTL.

Non-Data-Transmitting Operations on Output Units

The IOBS allows for several non-data-transmitting operations that may be used with output files.

Backspace: To obtain a backspace over a single block, the programmer can use the following calling sequence:

```
TSX    S.BSR,4
PZE    file,,bof
```

The symbol *bof* is the location of a routine to be transferred to if the block count in the file control block is zero.

When this calling sequence is given, the current buffer is truncated and written out, and the unit is backspaced over the record just written. The result of a backspace operation on a mixed-mode file is unpredictable.

Force End-of-Reel: If the programmer wishes to simulate the end of medium on an output file, he can use the following calling sequence:

```
TSX    S.FEOR,4
PZE    file
```

When this calling sequence is used, the current buffer is truncated and written out. A file mark is written, and if the file is labeled, a trailer label indicating the end-of-reel condition is created and written, and another file mark is written. The unit is then rewound and unloaded, and primary and secondary units are switched. The secondary unit is rewound. If the system is labeled, the retention period of the old header label is checked. If the file is labeled, a new header label is written, followed by a file mark. If the file control block indicates that a checkpoint should be written when units are switched (either on the new unit of the file or on the checkpoint file), this checkpoint is written. The job is terminated if the file is unopened.

Write File Mark: If the programmer wishes to write a file mark on an unlabeled output file, he can use the following calling sequence:

```
TSX    S.WEF,4
PZE    file
```

This calling sequence causes the current buffer to be truncated and written out and a file mark to be written following it. If an end-of-medium indicator is detected while the file mark is being written, the end-of-reel exit specified by the file control block is taken. If the file specified by this calling sequence is not open, is a labeled output file, or is any input file, no operation is performed and the job is terminated.

Rewind: If the programmer wishes to rewind an unlabeled file, he can use the following calling sequence:

```
TSX    S.REW,4
PZE    file
```

This calling sequence causes the current buffer to be truncated and written out and the unit to be rewound. If the calling sequence is given for an unopened file or for a labeled file, no operation is performed and the job is terminated.

Label Handling

The following operations may occur with output files:

Header Label: When a labeled or unlabeled output file is opened and at the beginning of each new reel of the file, if the system is labeled, the retention period specified by the header label of the file already on the reel is checked. [Labeled systems are described in "Output Unit Header Label" under "Input/Output Label System (IOLS)."] If this retention period has expired, the reel is accepted, and if the file is labeled, a header label for the new file is created and written. If the retention period has not expired, a pause for operator intervention occurs. The operator may cause the reel to be accepted, mount a reel that has an expired retention period (this one is also verified), or terminate the job.

Trailer Label: When the end-of-medium indicator is sensed or when a call is made to S.FEOR, if the file is labeled, an end-of-reel trailer label is created and written. Units are then switched. An end-of-file trailer label is formed and written when the file is closed, if the file is labeled.

Closing

When the processing of a file has been completed, the file should be closed with the following calling sequence:

```
        TSX       S.CLSE,4
        pfx       file
```

where:

pfx
S bit = 0 if the unit is to be rewound.
= 1 if the unit is not to be rewound.
1 bit = 0 if the unit is not to be unloaded when it is rewound.
= 1 if the unit is to be unloaded when it is rewound.
2 bit = 0 if the end-of-data-file procedure is to be executed.
= 1 if the end-of-data-file procedure is not to be executed.

The end-of-data-file procedure on labeled output files consists of creating and writing an end-of-file trailer label, preceded and followed by file marks. For an unlabeled output file, the end-of-data-file procedure consists of writing a file mark. There is no end-of-data-file procedure for input files.

When a file is closed, the following sequence of actions occurs:

1. If the file control block shows that the file was not open, only the rewind operations specified in the calling sequence are performed. Control is then immediately returned to the calling program.
2. If the file is an input file and the MON option is specified in the calling sequence, the file is unbuffered in closing. That is, the unit to which the file is attached is positioned in the correct location to read the block following that from which the last logical record requested by the user was obtained.
3. If the file is an output file, the current buffer is truncated and written out.
4. The rewind and end-of-data-file options specified in the calling sequence are executed.
5. The buffers are returned to their pool.
6. The OB (Open Status) bit in the file control block is turned off.

The result of closing and re-opening a mixed-mode file is unpredictable.

If it is desired to close several files at a time, the following calling sequence can be used:

```
        TSX       S.CLSL,4
        PZE       a,t,n
```

where:

a,t
is the location of the first of a series of words of the same form as word 1 of the S.CLSE calling sequence, each defining one of the files to be closed and the actions to be taken for that file.

n
is the number of files to be closed.

Miscellaneous Considerations

Nonstandard and Additional Label Processing

The IOBS allows the user to create and verify labels different from those usually provided, and still use the automatic unit switching and label writing and reading facilities. This is done by specifying either nonstandard or additional label processing in the file control block.

If nonstandard label processing is specified for an input file, IOBS reads each label when a label should appear on the unit and then transfers control to the nonstandard processing routine specified in the file control block. Similarly, when an output file has nonstandard labels, IOBS provides the nonstandard routine with a 20-word buffer in which to create a label and then writes the buffer out when control is returned to it.

If additional label processing is specified, IOBS performs all of the verification and creation that would normally be performed and then transfers control to the user's routine for the verification or creation of extra fields in the 20-character optional use field of the standard label. The user's routine is not entered if an error in the label is detected by IOBS and the operator takes the option of accepting the reel.

For complete information on this subject, the reader should refer to the sections "Format of Labeled Files," "Format of Labels," and "Nonstandard and Additional Label Processing" in the chapter "Input/Output Label Systems (IOLS)." It is with this part of IOCS that the user's nonstandard and additional label processing routines communicate.

Unlabeled Files

The user may specify an unlabeled file as being either single-reel or multi-reel. This specification affects the action taken by IOBS when a file mark is read.

When a file mark is read on a single-reel unlabeled input file, IOBS transfers control to the end-of-file routine specified in the file control block (described in the section "End-of-File Exits").

When a file mark is read on a multi-reel input file, IOBS transfers control to the end-of-reel routine specified in the file control block. This routine can either close the file or return control to IOBS.

In the latter case, automatic reel switching occurs and control is returned to the main program. It should be noted that the end-of-file exit is never used on an unlabeled multi-reel file. It is therefore necessary that the user provide his program with some other means of detecting the end of the data file.

The end-of-reel exit is taken whenever an end-of-medium indicator is sensed on an output unit, regardless of whether or not the output file is multi-reel. The end-of-reel routine can close the file, return control to IOBS to cause unit switching, or continue to write on the unit. The last possibility should be avoided, because continued writing may result in job termination if the symbolic unit being written on is disk storage or may pull the end of the tape off the reel if the symbolic unit is a magnetic tape unit.

Multi-File Reels

It is possible for a given input/output unit to hold more than one file. To process these files, the user simply opens the first file, processes it, closes it without a rewind, opens the next file without a rewind, etc. If the files are labeled, the closing procedure on each file leaves the unit positioned correctly for opening the next file (i.e., just before the header label).

End-of-Reel Exits

Whenever an end-of-reel trailer label is read, a file mark is read on an unlabeled multi-reel input file, or the end-of-medium indicator is detected on an output unit, the following calling sequence is executed by IOBS:

```
        TSL       eorex
```

where:

eorex
is the location of the user's end-of-reel routine for the file, specified in the file control block.

When this exit is taken, the accumulator contains the following information:

```
        PZE       file
```

If unit switching is desired, control may be returned to IOBS by a TRA* eorex.

End-of-File Exits

When an end-of-file trailer label is read, when a file mark is read on a single-reel input file, and when the user attempts to GET a record or block from an unopened input file, the following instruction is executed:

```
        TRA       eofex
```

where:

eofex
is the location of the user's end-of-file routine, specified in the file control block.

When this exit is taken, the accumulator contains the following information:

```
        PZE       file
```

IOBS Error Procedures

The IOBS allows the user to provide a routine of his own for taking corrective action on the following kinds of errors:

ERROR NUMBER	CONDITION
1	Block sequence number errors
2	Check sum errors
3	Both block sequence and check sum errors
4	An unrecoverable read error
5	An attempt to write on an unopened output file
6	Buffer overflow (an attempt to write a data record longer than the buffers attached to the file)
7	Reading of a Type 2 or Type 3 record in which the control word specifies a record length greater than the number of words remaining in the buffer, or a request to GET a Type 1 record longer than the buffers attached to the file
8	Unexpected mode change
9	An unrecoverable write error
10	Incomplete word read

When any one of the above errors is encountered, IOBS executes the following instruction:

```
        TSL       errex
```

where:

errex
is the location of the entry point to the user's error-correction routine, specified in the file control block.

The following information is left in the accumulator:

```
        PZE       file,,n
```

where:

n is one of the error numbers shown.

All three index registers have the same contents that they had when IOBS was entered.

The error-correction routine can restore all index registers and return control to IOBS by a TRA* errex. If this is done, the following actions are taken for the various kinds of errors:

a. If the error was of type 1, 2, 3, 4, 8, 9, or 10, the block in error is accepted as valid.

b. If the error was of type 5, the file control block is tested to see if the file is now open. (The error routine can call S.OPEN for the file.) If the file is open, processing continues; if it is not, the job is terminated.

c. If the error was of type 6, IOBS processes as much of the block as will fit in the buffer and ignores the rest.

d. If the error was of type 7, IOBS sets the logical record length field in FCLNG to the number of words remaining in the buffer.

The user can accept an error record for error types 1, 2, 3, 4, 7, 8, or 10 by returning control to IOBS by a TRA* errex and initiating a request for another record when IOBS returns control to his program.

The user can ignore input file error returns for error types 4 and 10 by continuing with a GET request within the file rather than returning control to IOBS with the TRA* instruction. If the file contains blocked records, ignoring the error condition results in losing all the records within the block.

When IOBS encounters one of the following errors, the job is terminated:

ERROR NUMBER	CONDITION
11	Caller returned to IOBS from error number 5 without opening the file
12	Insufficient buffers to open this file
13	No end-of-file exit provided for this file
14	An attempt to GET or PUT a record on the checkpoint file
15	An attempt to PUT a record on an input file
16	An attempt to GET a record on an output file
17	An attempt to perform an invalid non-data operation
18	An attempt to write a file mark on an input file
19	An attempt to switch reels on a single-reel output file

Input/Output Operations (IOOP)

Some applications involve data processing methods that do not require the facilities of IOBS or that cannot be implemented using IOBS. For this reason, the structure of IOCS has been designed to allow the programmer to use the Input/Output Operations level of IOCS directly.

Input/Output Operations (IOOP) performs the reading and writing operations of IOCS. It contains facilities for reading and writing physical records and performing non-data-transmitting operations with the devices supported. It also provides an automatic response to real-time-device input, e.g., teleprocessing and telegraph messages, and includes a buffer-chain manipulation facility for input on these devices.

A synchronization routine, a calling sequence interpretation routine, and a select and error-recovery routine for each device are contained in IOOP. The synchronizer coordinates the initiation and completion of input/output operations with the execution of the user's program. It allows four different modes of synchronization to be used.

The calling sequence interpreter accepts requests made by the main program for input/output operations. Because requests for operations are recorded in the system control block that corresponds to the symbolic unit being used, requests cannot be stacked by IOOP.

The select and error-recovery routine for each device translates the request for an operation made by the user's program into the 7040 input/output instructions necessary to perform the operation on the unit specified. If there is an error when the operation is performed, the select and error-recovery routine initiates measures to correct (i.e., *recover*) the error.

A special form of the call to IOOP for telecommunications *input* causes the device to be "opened." This call establishes linkages to the system control block, unit control block, buffer chains, and special routine. Data transmission takes place only if an attention trap occurred prior to this call. Once the device is opened the attention traps initiate the steps by which the messages are read and placed in a buffer chain. Devices that have been "opened" may also be "closed," e.g., when all messages have been processed and no more are expected.

Telecommunications *output* units are treated the same as unit-record equipment. To initiate output for teleprocessing devices or telegraph units, the user must make a call to IOOP for each message to be written. A synchronized call for an output operation will wait for the attention that signals the completion of message transmission. When required by a device, the formats and special character codes must be furnished by the user.

Use of IOOP

The use of the IOOP level of IOCS requires the following actions on the part of the programmer:

1. He must reserve the buffers necessary for his application.
2. He must write into his program any coding necessary for buffer supervision or for blocking and deblocking data records. For telecommunications input devices, he must write a *special routine* that contains any coding necessary to locate the message in a "filled buffer chain," unlink the buffer, process the message, and return the empty buffer to an "available buffer chain." He must also provide error procedures within the special routine, since the normal error return is never taken by IOOP for data-transmission errors on these devices.
3. In the parts of his program that require the reading or writing of records or some non-data-transmitting operation on an input/output unit, he must use a calling sequence to IOOP. This calling sequence specifies the operation to be performed, the symbolic unit to be used, and the method of synchronization required. He may also specify the location and priority of a special routine to be executed as soon as the operation has been completed. For telecommunications input devices, he must provide a calling sequence to "open" each device. This calling sequence supplies the location of control words for both the "available buffer chain" and the "filled buffer chain." It specifies the symbolic unit to be used, and the entry point and priority of a special routine (required for these devices).

Buffers

When reserving space for buffers into which records will be read and from which records will be written by IOOP, the user should reserve an area equal in length to the size of the largest possible physical record, plus two. One extra word, at the beginning of the buffer, is used by IOOP as a buffer control word. When a request for a read or write operation is made, the parameter given to IOOP should specify the location of the first data word to be read or written, i.e., the word immediately following the buffer control word. Each input buffer should also have an extra word at the end, to allow for the possibility of a device error. When a request has been made to IOOP to have a record read into or written from a given buffer, no reference can be made to the buffer or to any word in it until the operation has been completed and checked.

IOOP contains, in the select routines for real-time input devices, an automatic buffering function that permits immediate response to teleprocessing device or telegraph unit attentions. Empty buffers are linked in an available buffer chain. When an input attention trap occurs, IOOP unlinks a buffer from the available chain, causes a message to be read into it, and links it to a filled buffer chain. IOOP also provides a routine that can be called by the user to locate the message in the filled chain, unlink the buffer, and return the empty buffer, after processing, to the available chain. (See "Buffer Chain Manipulation Routine.") The user must provide the routine that processes the message.

Symbolic Units Table

When the Nucleus is assembled, one word of core storage is reserved for each symbolic unit defined. These words form the Symbolic Units Table. The table is used to record the correspondence between symbolic units and physical devices, and the status of each unit. When a symbolic unit has been attached to a physical device, the format of the Symbolic Units Table entry is:

S.Sxxx

0 RS 0	l(scb)	000	l(ucb)
S 2	3 17	18 20	21 35

where:

S.Sxxx
is the name of the symbolic unit.

RS
= 0 if this unit is not in use by an object program.
= 1 if this unit is in use by an object program.

l (scb)
is the location of the system control block for this symbolic unit.

l (ucb)
is the location of the unit control block for the input/output device to which this symbolic unit is attached.

The name of the symbolic unit is the symbol for the location of this entry in the Symbolic Units Table, and is recognized as such by the Macro Assembly Program (IBMAP). System control blocks are discussed later in this chapter. Unit control blocks are used to record information concerning the various input/output devices attached to the 7040/7044, and each one contains the machine address of the physical device to which it corresponds. Unit control blocks are discussed in the chapter "Input/Output Executor (IOEX)."

System Control Blocks

In addition to the Symbolic Units Table, IOOP requires a system control block for each symbolic unit. For 1402, 1403, 1622, 729, and 7330 devices, the system control block is four words long. For 1009, 1011, 1014, and telegraph devices, attached through a 1414, Model IV or V, Input/Output Synchronizer, the system control block is four words long. For a symbolic unit that consists of a section of 1301 Disk Storage or 7320 Drum Storage, the system control block is eight words long. The last four words apply only to devices requiring an order.

The format of the system control block is shown in Figure 7.

The possible types of system control blocks, as specified in field SCTYP, are:

00	Not specified
01	Printer on channel A (1403)
02	Card punch on channel A (1402)
03	Card reader on channel A (1402)
04	Magnetic tape unit on channels A-E
05	Paper tape reader on channel A (1011)
06	Card punch on channel A (1622)
07	Card reader on channel A (1622)
10	Not specified
11	Printer on channel S (the on-line 1401)
12	Card punch on channel S
13	Card reader on channel S
14	Magnetic tape unit on channel S
15-17	Not specified
20	Disk or drum storage, accessed randomly
21	Disk or drum storage, accessed sequentially in single-record fashion
22	Disk or drum storage, accessed sequentially in full-track with addresses mode
23	Disk storage, accessed sequentially in cylinder mode
24	Drum storage, accessed sequentially in cylinder mode
25-40	Not specified
41	Data transmission unit on channel A, output (1009)
42	Remote inquiry unit on channel A, output (1014)
43	Telegraph unit on channel A, output (TTY)
44	Not specified
45	Data transmission unit on channel A, input (1009)
46	Remote inquiry unit on channel A, input (1014)
47	Telegraph unit on channel A, input (TTY)
50-77	Not specified

The possible contents of the reserve status field, SCRES, are as follows:

00	Unreserved and not in use
01-62	Intersystem reservation requested by the object program
63-67	Not specified
70	Unreserved and in use by the object program
71	Unreserved and in use by a priority program

<table>
<tr><th></th><th>S</th><th>1</th><th>2</th><th>3 4 5</th><th>6 7 8 9 10 11</th><th>12 13 14 15 16 17</th><th>18 19 20</th><th>21 22 23 24 25 26 27 28 29 30 31 32 33 34 35</th><th></th></tr>
<tr><td>1. SCSIZ
SCTYP
SCRES</td><td colspan="4">Length of this system control block</td><td>Type of this system control block</td><td>Reserve status of this symbolic unit</td><td>▨</td><td>Location of the word in the Symbolic Units Table that points to this system control block</td><td>SCUNI</td></tr>
<tr><td>2. SCOPR</td><td colspan="6">Operation currently requested for this unit *</td><td>▨</td><td>Location of the standard select and error-recovery routine for this unit</td><td>SCSEL</td></tr>
<tr><td>3. SCIOC</td><td colspan="8">The input/output channel command is placed in this word when the operation is requested,* and the store channel and error information is left in this word when a file mark is read.**</td><td>SCSCH</td></tr>
<tr><td>4. SCSNS</td><td>E M*</td><td>B M **</td><td></td><td colspan="3">Sense status information for devices requiring an order **</td><td>PR *</td><td>Location of the entry point to the special processing routine specified for this operation *</td><td>SCRTN</td></tr>
<tr><td>5. SCLIM</td><td colspan="3"># of recs per track (1st 3 bits)</td><td colspan="3">Track number of the first track assigned to this symbolic unit</td><td># of recs per track (last 3 bits)</td><td>Track number of the last track assigned to this symbolic unit</td><td></td></tr>
<tr><td>6. SCPOS</td><td>S R</td><td colspan="2">▨</td><td colspan="3">Track address of next record if cylinder mode, otherwise record number of last file mark**</td><td>▨</td><td>Track number of the last file mark written on the unit **</td><td></td></tr>
<tr><td>7. SCDO1</td><td colspan="5">Disk order *</td><td>Current record number **</td><td>▨</td><td>Current track number *</td><td></td></tr>
<tr><td>8. SCDO2</td><td>W C*</td><td>N R*</td><td colspan="6">Verification address of this record *</td><td></td></tr>
</table>

* Inserted by the IOOP unit synchronizer.
** Inserted by the IOOP select and error-recovery routine.

The meanings of the two-character codes are:

EM - End-of-Medium Indicator
= 0 if no end-of-medium indicator has been detected.
= 1 if an end-of-medium indicator has been detected.

BM - Beginning-of-Medium Indicator
= 0 if the device is not positioned at the beginning of the medium.
= 1 if the device is positioned at the beginning of the medium.

PR - Priority Indicator
is the priority of the special routine whose entry point address is held in the address portion of word SCRTN.

SR - Split Record
= 0 current record is not split.
= 1 current record is split between cylinders.

WC - Write Check Indicator
= 0 if no write check is to be performed.
= 1 if a write check is required.

NR - No Record Found Bit.
= 0 if the standard procedure is to be performed for a no-record-found condition.
= 1 if the special procedure is to be performed for a no-record-found condition.

Figure 7. Format of the System Control Block

72	Unreserved and in use by a permanent program
73	In use as a S.SLBx unit
74	In use as a S.SINx unit
75	In use as a S.SOUx unit
76	In use as a S.SPPx unit
77	In use as a S.SCK1 unit

The octal codes marked as not specified are reserved for future DS Programming Systems use.

Object programs must respect the reservation status of symbolic units. The reservation status of the unit is not checked by IOOP when an operation is requested.

The system control block is used by IOOP to retain information about each opeartion requested on the symbolic unit. The calling sequence interpreter provides the initial entries in the various fields, but the information is used and updated by the select and error-recovery routines. Any synchronizer for the IOEX level that uses the standard select and error-recovery routines of IOOP must perform this initialization function.

IOOP Calling Sequence

Only one entry point is used by IOOP. The parameters of the calling sequence describe the operation, the unit, and the input/output area to be used. One generalized calling sequence is used for all data-transmission requests. A slightly modified form of this calling sequence, described separately, is used for the "opening" and "closing" of a telecommunications input device. The format of the calling sequence to IOOP for data-transmission is:

```
TSX     S.IOOP,4
oper    ioparm,t1
pfx1    S.Sxxx,t2
pfx2    erret,t3,sprtn
```

where:

oper
is an operation code describing the operation to be performed (e.g., read or write) and the method of synchronization to be used. The assembled form of this operation code extends into the decrement portion of the word.

ioparm,t1
is the location of a word of the following form:

0 0 0	count	0 0 0	first word
S 2	3 17	18 20	21 35

where:

count
is the number of words to be read or written. If *count* is zero, this has the same effect as IODLY, and *sprtn* is ignored.

first word
is the location of the first word of the buffer into which data is to be read or from which data is to be written.

pfx1
= PZE if this word is the last one in the calling sequence (if no error return or special routine is specified).
= MZE if this word is not the last one in the calling sequence.

S.Sxxx,t2
is the location of a word specifying a system control block and a unit control block. This word may be a Symbolic Units Table entry, or it may be a word constructed by the programmer.

pfx2
= PZE, PON, PTW, PTH, or MZE if the entry point of a special routine of priority 0, 1, 2, 3, or 4 is specified in the decrement portion of this word.

erret,t3
is the location of the entry point to an error routine. The error routine is entered if an error occurs during the operation synchronized by this calling sequence. If an error occurs during the operation and no error return has been specified, the job is terminated. The use of the error return is discussed in "Status Information."

sprtn
is the location of the entry point to a special processing routine that is to be entered when the operation initiated by this calling sequence has been completed. Special processing routines are discussed in "Special Routines."

Index registers 1, 2, and 4 are saved when this calling sequence is executed, and they are restored before control is returned to the calling program. Locations S.SSCH and S.SSNS are set for the operation synchronized by this calling sequence. Multiple tagging is allowed in the tag fields of the calling sequence, but it should be remembered that multiple tags result in adding to the specified address the 2s complement of the logical OR of the contents of the index registers used.

The format of the calling sequence to open a telecommunications input device is:

```
TSX     S.IOOP,4
IOTPI   ioparm,t1
MZE     S.Sxxx,t2
pfx2    erret,t3,sprtn
```

where:

ioparm,t1
is the location of a word of the following form:

0 0 0	fchain	0 0 0	achain
S 2	3 17	18 20	21 35

where:

achain
is the address of the control word (BPCHN) for a chain of available buffers (see "Assembling Buffer Chains").

fchain
is the address of the control word (BPCHN) for a chain of filled buffers.

The other parameters of this calling sequence are the same as the corresponding ones for data transmission. The parameter sprtn is required. The normal error return is taken for program errors only.

The format of the calling sequence to close a telecommunications input device is identical to that for opening the device, except that ioparm,t1 is the location of a word of the form:

```
PZE     0,,-1
```

A call to IOOP to open a teleprocessing or telegraph input device should not be made if the device has already been opened. A device should not be closed until the special routine has processed all messages, and no further messages are expected.

A method has been provided to allow indirect addressing of the words of the IOOP calling sequence. To address a given word of the calling sequence indirectly, the programmer uses an IOOP calling sequence in which the indirectly addressed word is replaced with a word of the following form:

```
pfx     loc,t,-1
```

where:

loc,t
is the location of the word that is actually to be used in this position in the calling sequence.

-1
(assembles as an octal 77777) indicates that the word is indirectly addressed.

Indirect addressing can extend to any depth. The only part of the IOOP calling sequence that cannot be indirectly addressed in this manner is *pfx1*, which must appear in its normal position in the calling sequence. An example of the use of indirect addressing to write the same record on two different output units is shown in Figure 8.

```
          TSX       S.IOOP,4                         $WRITE RECORD ON UNIT 1
PARM4     PZE       PARM1,,-1
          PZE       PARM2,,-1
          TSX       S.IOOP,4                         $WRITE RECORD ON UNIT 2
          PZE       PARM4,,-1
          PZE       PARM3,,-1
          ...       .......
          ...       .......
PARM1     IOWDS     IOCOM
PARM2     PZE       S.SU01
PARM3     PZE       *+1,,-1
          PZE       S.SU02
IOCOM     PZE       FWORD,,6
          PZE       **                               BUFFER CONTROL WORD
FWORD     BCI       6,THIS IS INDIRECT ADDRESSING
```

Figure 8. Example of the Use of IOOP

Synchronization Types

In using IOOP, the programmer may specify one of the following four types of synchronization:

Priority
Synchronize
Initiate
Return Immediate

The method of synchronization chosen determines the point in the progress of the operation at which control is returned to the calling program and also affects the treatment of error returns and special processing routines.

IOOP does not allow for synchronization of teleprocessing or telegraph *input* devices. All input activity on these devices occurs as a result of an externally initiated attention, and not as the direct result of a call to IOOP.

PRIORITY REQUEST

When the calling sequence to IOOP indicates that Priority synchronization is required, the following sequence of actions is performed:

1. If an operation is already in progress on the specified unit, IOOP waits until that operation has been completed and checked, and until any necessary special processing routine has been executed. Such a previous incomplete operation is probably the result of a logical error, because, if an error return is necessary, the error return specified by *this* calling sequence (i.e., for the Priority operation) is used and the Priority operation is neither recorded nor initiated. No distinction is made in the error return between an error caused by the previous operation and one caused by the Priority operation.

2. The request for the new operation is recorded and a request is made to IOEX for a Priority operation on the channel.

3. The IOOP waits for the completion of the operation and the execution of any special processing routine that may be required.

4. If an error return is required, control is transferred to the error routine specified in the calling sequence for this operation.

5. If the error return is not required, control is returned to the calling program.

The operations that use Priority synchronization are:

IORBP	Read binary data with priority.
IORDP	Read BCD data with priority.
IOWBP	Write binary data with priority.
IOWDP	Write BCD data with priority.

SYNCHRONIZE REQUEST

When the calling sequence indicates that the Synchronize method of synchronization is to be used, the following sequence of actions is performed:

1. If an operation is already in progress on the specified unit, IOOP waits until that operation has been completed and checked, and until any necessary special processing routine has been executed. Such a previous incomplete operation is probably the result of a logical error, because, as was the case in Priority operations, the error return for this calling sequence is used if an error was found in the previous operation and the new operation is neither recorded nor initiated.

2. The request for the new operation is recorded and a request is made to IOEX for an operation on the channel.

3. The IOOP waits for the completion of the operation and for the execution of any special processing routine that may be required.

4. If an error return is required, control is transferred to the error routine specified in the calling sequence for this operation.

5. If the error return is not required, control is returned to the calling program.

The operations that use the Synchronize method of synchronization are:

IORBS	Read binary data and synchronize.
IORDS	Read BCD data and synchronize.
IOWBS	Write binary data and synchronize.
IOWDS	Write BCD data and synchronize.

The only difference between Priority and Synchronize requests for a unit is that the Priority request results in a request to IOEX for priority use of the channel to which the unit is attached.

INITIATE REQUEST

When the calling sequence indicates that the Initiate method of synchronization is used, the following sequence of actions is performed:

1. If an operation is already in progress on the specified unit, IOOP waits until that operation has been completed and checked and until any necessary special processing routine has been executed. If an error return is necessary, the error return specified by this calling sequence is used and the new operation is neither recorded nor initiated.

2. The request for the new operation is recorded and a request is made to IOEX for an operation on the channel.

3. Control is returned to the calling program.

The operations that use Initiate synchronization are:

IORBI	Read binary data and initiate.
IORDI	Read BCD data and initiate.
IOWBI	Write binary data and initiate.
IOWDI	Write BCD data and initiate.

RETURN IMMEDIATE REQUEST

When the calling sequence indicates that the Return Immediate method of synchronization is to be used, the following sequence of actions is performed:

1. If an operation is already in progress on the specified unit the sign of the word specified by *ioparm, t1* in the calling sequence is set to minus and control is returned to the calling program.

2. If no operation is in progress, but a previously completed operation requires an error return, the error return specified by this calling sequence is used and the new operation is neither initiated nor recorded.

3. If the unit is not busy and no error return is required by a previous operation, the request for the new operation is recorded and a request is made to IOEX for an operation on the channel.

4. Control is returned to the calling program.

The operations that use Return Immediate synchronization are:

IORBR	Read binary data or return immediate.
IORDR	Read BCD data or return immediate.
IOWBR	Write binary data or return immediate.
IOWDR	Write BCD data or return immediate.

The only difference between Initiate and Return Immediate requests for a unit that IOOP does not wait for a unit to become free when a request is for Return Immediate. The user must check any Return Immediate request to see if it has been accepted, and if not, he must repeat the request.

Special Routines

When a calling sequence to IOOP requests an operation and the execution of a special routine, the location and priority of this routine are stored in the system control block. When the IOOP select and error-recovery routine for the device being used finds that the operation has been completed, it requests the execution of the special routine by placing its address and priority in the unit control block. After another input/output operation has been initiated on the channel, if possible, IOEX looks for the higest priority routine waiting to be executed, places PZE l(ucb) in the accumulator, and enters the special routine by a TSX sprtn,4.

Because the special routine is entered before any error routine, it is possible for the special routine to be entered for an operation that resulted in a permanent error. The special routine should examine the results of the operation, in words SCSCH and SCSNS of the system control block, to determine whether or not an error has occurred. If the special routine stores PZE 0,,1 into word SCSCH, no error return will be taken.

Trapping is not inhibited during the execution of a special routine, but all registers are saved when any program is trapped. If a special routine of higher priority than the one in progress is requested during trap-time, this routine is executed before the interrupted routine is re-entered. Otherwise, the registers are restored and the interrupted routine is re-entered as soon as the treatment of the trap is complete.

A special routine may have any priority from zero to four, as indicated by the prefix of the last word of the calling sequence to IOOP that requests it. The main program has no priority.

Buffer Chains

In order that teleprocessing or telegraph messages can be accepted promptly when an attention trap occurs, two buffer chains must be provided for each telecommunciations input device. The select routine for these devices unlinks a buffer from the available buffer chain, and the message is read into the buffer. The location of the buffer is inserted in field SCOPR of the system control block. The select routine then places the status information for the device into the buffer flag word (BPFLG) preceding the now-filled buffer (field SCSCH of the system control block is not used) and links the buffer to the filled buffer chain. The select routine then requests scheduling of the special routine, and an MZE is placed in field SCOPR of the system control block.

If another attention trap occurs before the special routine has been completed, the select routine unlinks another buffer from the available buffer chain, causes it to be read into, links it to the filled buffer chain, and returns control to the special routine. Thus, a special routine may find more than one buffer in the filled buffer chain. If multiple interrupts of the special routine have occurred, the routine will be re-entered because of a request posted for it by the last attention trap, and it will find that the filled chain may have already been emptied.

While each teleprocessing or telegraph input device must have an available buffer chain and a filled buffer chain, these need not be unique for each device. A special routine may in fact return a buffer to an available chain other than the one from which it came. It is the responsibility of the user's special routine to ensure that each device's available buffer chain is replenished, and that sufficient buffers are available for the maximum number of anticipated messages before the chain can be replenished.

If the select plus routine finds that a buffer is not available, the sign of the buffer-length word (BPLGN) of the available chain control-word pair is set to minus. The routine then transfers to IOEX with a 2, 4 return, requesting rescheduling. If the available buffer chain is not replenished in time, one or more messages may be lost.

The pointers to the two chains appear in word SCIOC of the system control block. This word may be assembled with the following:

```
              PZE        achain,,fchain
```

ASSEMBLING BUFFER CHAINS

The buffer-chain control words, the associated buffer chains, and their buffers must be assembled such that they will occupy a portion of core storage that can never be overlaid by a background program. A buffer-chain control-word pair for an available chain may be assembled as:

```
ACHAIN     MZE      NEXTn,,NEXT1      (BPCHN)
           PZE      ,,lgn             (BPLGN)
```

An available buffer chain may be assembled as:

```
NEXT1      PZE      BUFF1,,NEXT2      (BPCON)
NEXT2      PZE      BUFF2,,NEXTn      (BPCON)
NEXTn      PZE      BUFFn,,ACHAIN     (BPCON)
           BSS      1                 (BPFLG)
BUFF1      BSS      lgn
           BSS      1                 (BPFLG)
BUFF2      BSS      lgn
           BSS      1                 (BPFLG)
BUFFn      BSS      lgn
```

where:

lgn
defines the length of the buffer.

No buffers need be provided for the filled chain. A buffer-chain control-word pair for an empty chain may be assembled as:

```
FCHAIN     MZE      FCHAIN,,FCHAIN (BPCHN)
           MZE      ,,lgn          (BPLGN)
```

BUFFER-CHAIN MANIPULATION ROUTINE

A routine (S.TPBF) is provided at the IOOP level to locate a message in a buffer chain and remove the buffer from the chain, and to return the buffer to a chain. Following are explanations for the use of these two functions:

(1) To locate a message and remove the buffer from a chain, the user must first place the following in the AC:

```
           PZE      PCHAIN
```

where:

PCHAIN
is the control word of the buffer chain from which the first buffer will be removed.

The format of the calling sequence to the routine is:

```
           TSX      S.TPBF,4
```

Upon return, the AC contains:

```
           PZE      ,,NEXTi
```

where:

NEXTi
is the location of the first buffer control word in PCHAIN (if NEXTi=0, there is no buffer in PCHAIN).

The MQ contains:

```
           PZE      BUFFi,,lgn
```

where:

BUFFi
is the location of the first buffer.

lgn
is the length of the buffer.

Before processing the buffer, the user should save the contents of the AC returned by the routine. This information is placed in the MQ when the buffer is returned to a chain.

(2) To return the buffer specified by NEXTI to a buffer chain, the user must place the following in the AC:

```
           MZE      ,,QCHAIN
```

where:

QCHAIN
is the control word of the buffer chain to which a buffer will be returned.

The MQ must contain:

```
          PZE         ,,NEXTi
```

The format of the calling sequence to the routine is:

```
          TSX         S.TPBF,4
```

While the S.TPBF routine is provided, its use is not required. If it is not used, the buffer chains should be manipulated as follows:

Assuming there is a buffer in the filled chain, the buffer chains should be presented to the user's special routine as:

```
FCHAIN    MZE     NEXT1,,NEXT1
          PZE     ,,LGN
ACHAIN    MZE     NEXTN,,NEXT2
          PZE     ,,LGN
NEXT1     PZE     BUFF1,,FCHAIN
NEXT2     PZE     BUFF2,,NEXTN
NEXTN     PZE     BUFFN,,ACHAIN
```

The special routine detaches the buffer from its chain leaving:

```
FCHAIN    MZE     FCHAIN,,FCHAIN
          MZE     ,,LGN
ACHAIN    MZE     NEXTN,,NEXT2
          PZE     ,,LGN
NEXT1     PZE     BUFF1,,**
NEXT2     PZE     BUFF2,,NEXTN
NEXTN     PZE     BUFFN,,ACHAIN
```

When the special routine finishes processing, the buffer is attached to the other chain as:

```
FCHAIN    MZE     FCHAIN,,FCHAIN
          MZE     ,,LGN
ACHAIN    MZE     NEXT1,,NEXT2
          PZE     ,,LGN
NEXT1     PZE     BUFF1,,ACHAIN
NEXT2     PZE     BUFF2,,NEXTN
NEXTN     PZE     BUFFN,,NEXT1
```

Status Information

Two fixed locations S.SSCH and S.SSNS, are used by IOOP to hold information about the status of all operations with one exception—status information about telecommunications *input* operations is stored in the buffer flag word, BPFLG. When an operation is synchronized, information regarding conditions that occurred during the operation is stored in S.SSCH and S.SSNS. The address portion of S.SSCH contains the location of the last word of data transmitted, plus one, or is zero if data has not been transmitted. Bits are set in the prefix and decrement of S.SSCH for the following conditions:

S The record was successfully read in the opposite mode (BCD/binary) from that specified for the operation.

5 The calling sequence requested the transmission of one or two words (729/7330 Magnetic Tape Units only).

6 No record found (for 1301 and 7320 only).
Addressed station inoperative (1014 output only).
Excessive delay or invalid format (telegraph output only).

7 End-of-medium indicator was detected while writing.

9 Program error occurred.

10 External device error was encountered.

11 External device error was encountered and ignored.

12 Unusual end occurred (incomplete word was transmitted if the device was a 729/7330 Magnetic Tape Unit).

13 Recovered error.

14 Channel parity.

16 Permanent error.

17 The operation was completed. (For 1009 output, this bit is set only if the end-of-message indicator is received for the request.)

Conditions 5, 6, 7, 9, 10, 12, 14, and 16 cause error returns.

The information found in location S.SSNS is described separately for each device requiring an order.

If the end of the medium was detected during a write operation, bit 7 is set and the error return is taken. Until the unit is rewound or unloaded, bit 7 is set on for each subsequent operation on the unit, but the error return is not taken.

If a file mark is read, S.SSCH is set to zero and the normal return is taken. If a special routine is inspecting word SCSCH of the system control block, it will find the information usually found in S.SSCH, and bit 15 indicates the reading of the file mark.

Program error returns are given when the following conditions occur:

1. If the user requests a read operation that would violate storage protection.

2. If IOOP receives a read or write request for a unit that is busy from a nonstandard select and error-recovery routine.

3. If a file mark is read while spacing forward by records (see "Non-Data-Transmitting Operations" under "729/7330 Magnetic Tape Units").

4. If a request is made to backspace on a unit that is already at load point.

5. If a request is made for an undefined operation, e.g., writing binary information on a 1622, reading from a 1403, writing a file mark on a random access device, or backspacing a unit record device.

6. If the user attempts to exceed the storage capacity of a symbolic unit (e.g., writing past the end of a sequentially accessed section of 1301 Disk Storage, as delimited by the system control block for the symbolic unit).

7. If the system unit field is zero.

8. If any request other than open or close is made for a telecommunications input device or if a request is made to open a device that has already been opened.

Programmer errors 1, 2, 7, and 8 prevent entry into a special routine.

Status information defining the results of teleprocessing and telegraph input operations is presented to the user in the input buffer, and must be examined in the special routine. Since input operations are externally initiated, i.e., not as the result of a call to IOOP, no error return is taken. The input status information is stored in the decrement of the buffer flag word, BPFLG, which precedes the input buffer (see "Assembling Buffer Chains"). These bits are defined as:

BIT	DESCRIPTION
10	Transmission error
14	Word parity error
15	All or part of message lost
16	Channel check
17	End of message (1009)
21-35	Location of the System Units Table

IOOP makes one exception for these input devices. If any call other than open or close is made for teleprocessing or telegraph input devices or if an attempt is made to open an already opened device, IOOP sets the program error bit in S.SSCH and takes the standard error exit without scheduling the request.

Non-Data-Transmitting Operations

Aside from its reading and writing facilities, IOOP provides four generalized non-data-transmitting operations. These are:

IODLY
This operation delays processing until any previously requested operation on the specified unit has been completed and checked.

IOWEF
This operation causes IOOP to write a file mark (or to perform some analogous operation) on the specified unit. Initiate synchronization is used.

IOSKP
This operation causes IOOP to space forward or backward on the specified unit. Initiate synchronization is used.

IODER
This operation causes the typing of an error message and allows the termination of the job, if required.

UNIT DELAY

The following calling sequence may be used to delay processing until the end of an operation:

```
TSX      S.IOOP,4
IODLY
pfx1     S.Sxxx,t2
pfx2     erret,t3
```

All of the elements of this calling sequence are defined as they were for data transmission. If an error occurs in the operation for which the delay has been requested, the error return specified by this calling sequence is used.

Although the program requesting the delay is suspended until the operation is complete, previously requested special routines of a higher priority are still executed when the corresponding operations have been completed and store channel data for these operations has been obtained. Calls from these special routines for input/output operations are accepted by IOOP.

WRITE FILE MARK

The following calling sequence may be used to write a file mark on the specified unit:

```
TSX      S.IOOP,4
IOWEF
pfx1     S.Sxxx,t2
pfx2     erret,t3,sprtn
```

All of the elements of this calling sequence are defined as they were for data transmission. Since IOWEF is an Initiate operation, the error return specified by this calling sequence is taken if an error has occurred in a preceding Initiate or Return Immediate operation.

The actions taken by IOOP, if this operation is requested on a unit other than a magnetic tape unit, are specified in the section "Input/Output Devices."

REPOSITIONING

The following calling sequence may be used to reposition the specified unit:

```
TSX      S.IOOP,4
IOSKP    ctlwd,t1
pfx1     S.Sxxx,t2
pfx2     erret,t3,sprtn
```

All of the elements of this calling sequence are defined as they were for data transmission, except that *ioparm, t1* has been replaced by *ctlwd, t1*. The address *ctlwd, t1* is the location of a word of the following form:

```
pfx      r,,f
```

where:

pfx
= PZE if the repositioning is forward.
= MZE if the repositioning is backward.
= PON if the repositioning is to the location of a specific record in the current file.

If the prefix is PZE (forward spacing), *r* is the number of physical records to be spaced over and *f* is the number of files to be spaced over. Each file mark read is counted as a file. The *f* files are spaced first, and then *r* records.

If the prefix is MZE (backward spacing), either *r* or *f* must be zero. If *r* is nonzero, the unit is backspaced *r* records (even if *r* is greater than the current record count). If *f* equals 1, the unit is backspaced until the record count has been reduced to zero.

If the prefix is PON, *f* should be zero. The unit is backspaced or forward spaced until it is in position to read (or write) record *r* + 1.

If *f* is −1, the unit is rewound. If the prefix is PZE, rewinding alone is accomplished. If the prefix is MZE, the unit is rewound and unloaded.

If the control word is PZE or MZE 0,,0; IOSKP has the same effect as an IODLY and *sprtn* is ignored.

DEVICE ERROR CALL

If an input/output error that was not detected by IOOP is detected by the user's program (e.g., incorrect word count, incorrect internal label, or nonagreement of check sums), the following calling sequence may be used to indicate this fact:

```
        TSX       S.IOOP,4
        IODER     ercode,t1
        pfx       S.Sxxx,t2,messag
        PZE                  NOT SIGNIFICANT
```

where:

ercode,t1
is the location of a BCD word identifying the error discovered by the program.

messag
is the location of the first of a pair of BCD words that describe the error (e.g., INPUT FORMAT).

This calling sequence causes the following message to be typed:

(ercode,t1) (messag) ERROR ON S.Sxxx, nnnnn

The five octal digits *nnnnn* form the machine address of the device in error.

The prefix *pfx* determines the action taken by IOOP after the message has been typed. The possibilities are:

pfx
= PZE Control is returned to the calling program by a TRA 3,4.
= MZE Control is returned to the calling program by a TRA 4,4.
= PON or MON The job is terminated.

The last word of the calling sequence is not significant, because it is impossible for an error return to be taken or a special routine to be scheduled by IODER.

Input/Output Devices

A single, device-oriented, select and error-recovery routine is used by IOOP to perform all of the reading, writing, non-data-transmitting, error detection, and error-recovery operations necessary for each device. The following sections describe the operations allowed by these routines for the various kinds of input/output units.

729/7330 Magnetic Tape Units

The length of each record written on magnetic tape by IOOP is equal to the number of words indicated in the parameter specified by the user's calling sequence. If the record on tape is longer than the number of words specified for a read operation, transmission ceases when the required number of words has been read, but tape motion continues until the interrecord gap at the end of the record is reached. If the record being read contains fewer than the number of words specified, transmission ceases when the interrecord gap is reached. If the tape is being read or written through an on-line 1401, the size of the 1401 buffer limits the record size. The user can determine the number of words transmitted to a magnetic tape unit on channels A-E by examining S.SSCH. This method cannot be used for magnetic tape units attached to the on-line 1401, however, because the Store Channel information is erroneous for these devices.

NON-DATA-TRANSMITTING OPERATIONS

The following non-data-transmitting operations may be used with 729/7330 Magnetic Tape units:

1. Space the tape forward over *f* file marks and then over *r* physical records.
2. Space the tape backward to the beginning of this file.
3. Space the tape backward over *r* physical records.
4. Place the unit in position to read the $(r + 1)th$ record of the current file.
5. Rewind the tape.
6. Rewind and unload the tape.
7. Write a file mark on the tape.

SEQUENCES OF OPERATIONS

Certain combinations of reading, writing, and backspacing operations may result in errors. The table in Figure 9 shows the possible combinations and the results of each. In this table, R_n stands for "read record n," W_n stands for "write record n," B_n stands for "backspace over record n," Rew stands for "rewind the tape," and S stands for "write blank tape." (IOOP has no call for a "write blank tape" operation.)

No check for a valid sequence of operation is made by IOOP. It is the responsibility of the programmer to ensure that his program does not violate any of the rules given above.

ERROR CONDITIONS

The error conditions that may be encountered with magnetic tape and the actions taken to correct them are:

Tape Redundancy While Reading: If the record in error is not a noise record, i.e., is three words or longer, the tape is backspaced over the record and the record is reread. If there is still an error, this procedure is repeated nine more times. If this has no effect, if no noise records have been read previously, and if at least

Operation	Can be Performed	Remarks
Reading after Writing:		
1. W_nR_{n+1}	No	
2. $W_nB_nR_n$	Yes	
3. $W_nB_nR_nW_{n+1}$	No	A. The change from reading to writing causes stray bits in the interrecord gap. Record n+1 may be unreadable.
4. $W_nB_nR_nSW_{n+1}$	Yes (729 unit)	B. Only allowed if a noise record detection routine is used (as it is with IOOP).
	No (7330 unit)	Unerased tape longer than 17 characters remains in the interrecord gap.
5. $W_nB_nR_nR_{n+1}$	No	
Writing after Reading:		
1. R_nW_{n+1}	No	See Remark A.
2. R_nSW_{n+1}	Yes (729 unit) No (7330 unit)	See Remark B.
3. $R_nB_nW_n$	Yes (729 unit) No (7330 unit)	See Remark B.
Rewinding:		
1. $R_nRewR_1R_2 \ldots R_{n-1}R_n$	Yes	
2. $W_nRewR_1R_2 \ldots R_{n-1}R_n$	Yes	
3. $W_nRewR_1R_2 \ldots R_{n-1}$ R_nR_{n+1}	No	
4. $W_nRewR_1R_2 \ldots R_{n-1}$ R_nW_{n+1}	No	See Remark A.
5. $W_nSRewR_1R_2 \ldots R_{n-1}$ R_nSW_{n+1}	Yes (729 unit) No (7330 unit)	See Remark B.

Figure 9. Sequences of Operation on Magnetic Tape Units

three records have already been read, the tape is backspaced over three records, to pass the record in error under the tape cleaner. Ten more attempts to read the record follow this tape cleaner procedure. If, after ten tape cleaner procedures, i.e., 101 attempts to read the record, the record still has not been read properly, the error is considered permanent. This error is indicated to the calling program by an error return.

To avoid unnecessary rereads, an attempt is made to read the record in the opposite mode (i.e., BCD or binary) after three attempts to read the record in the original mode have failed. If this fourth attempt is successful, the record is accepted and an indication of the alternate mode is placed in s.sscH. If it is unsuccessful, the error-recovery procedure is continued in the original mode.

Tape Redundancy While Writing: The tape is backspaced over the record in error, and a second attempt is made to write this record. If this attempt is unsuccessful, the tape is backspaced again and blank tape is written. An attempt is made to write the record on the new piece of tape thus reached. If this attempt is unsuccessful, the tape is backspaced and blank tape is written again. After a total of 25 write-blank-tape operations, i.e., 27 unsuccessful attempts to write the record, the error is indicated to the calling program by an error return.

Incomplete Word While Reading: Two attempts are made to read the record. If the condition persists, it is indicated to the calling program by an error return.

Incomplete Word While Writing: The error is indicated to the calling program by an error return.

1301 Disk Storage or 7320 Drum Storage

The IOOP level of IOCS provides for system operation with either 1301 Disk Storage or 7320 Drum Storage; therefore, any reference to disk storage in the following section also applies to 7320 Drum Storage, unless otherwise stated.

IBM 1301 Disk Storage units can be used through IOOP in two distinct ways: sequential access and random access.

SEQUENTIAL ACCESS

When IOOP is used to read and write sequentially on disk storage, the track and record addresses specified in the system control block and unit control block are incremented automatically. The user who is writing sequentially or reading sequential records originally written by IOOP can program in essentially the same way that he would for magnetic tape, largely ignoring the special characteristics of disk storage. The major differences between sequential access use of disk storage and sequential use of magnetic tape are the record length limitations of disk storage and the symbolic unit limits. Sequential access can be used with three methods of operation: single record, full track with addresses, and cylinder mode.

The method of use of disk storage is specified in the system control block (described in the section "System Control Blocks"). The number of words that may be written in a given record, track, or cylinder is specified by the format track for the cylinder. If a record to be written exceeds this maximum, the extra words are not written (i.e., the record is truncated to the maximum possible size). Disk storage being read or written in cylinder mode must have a format of exactly 465 words per track. Drum Storage being read or written in cylinder mode must have a format of exactly 530 words per track. If an entire cylinder is not required for the block, the next track is used to begin the next cylinder operation. When formatting, one additional word should be provided in single-record and cylinder mode record areas to include the buffer control word. Full track operations write the buffer control word in the record address.

The end-of-medium bit in S.SSCH is set and the *erret* exit is taken to indicate that *n* or fewer tracks remain unused on the symbolic unit, where *n* is an installation assembly parameter. (The publication *IBM 7040/7044 Operating System (16/32K): Systems Programmer's Guide*, Form C28-6339, contains detailed information on installation assembly parameters.) The number of tracks that remain is determined by the mode of operation of the disk or drum. In full-track mode or single record mode, exactly *n* tracks remain; in cylinder mode, *n* or fewer tracks remain.

Normally *n* is set to 4, allowing the writing of an alternate buffer, a file mark, a trailer label, and another file mark. If this end-of-medium indication is ignored and the program continues to write on the unit, the program error return is taken when an attempt is made to write beyond the last track assigned to the unit.

NON-DATA-TRANSMITTING OPERATIONS

When disk storage is used sequentially, the following non-data-transmitting operations may be used:

1. Space forward over *f* file marks and then over *r* physical records (the physical record may be either a track or a single record, depending on system control block specifications).

2. Space backward to the beginning of this file.

3. Space backward *r* physical records.

4. Place the unit in position to read or write the (*r*+1)*th* record of the current file.

5. Rewind the unit (reposition to the beginning of this symbolic unit, as defined by the limits in the system control block).

6. Unload the unit. This operation causes an operator message to be typed.

7. Write a file mark on the unit. This operation causes the writing of a record that is recognizable to IOOP as a file mark when it is read.

RANDOM ACCESS

Facilities exist in IOOP for reading and writing on disk storage in random access fashion. When disk storage is used in this way, a slightly modified version of the standard calling sequence must be used. This calling sequence is:

```
TSX     S.IOOP,4
oper    ioparm,t1
pfx1    S.Sxxx,t2,order
pfx2    erret,t3,sprtn
```

The only difference between this calling sequence and the standard one is that the decrement of the third word (*order*) contains the address of the first of a pair of words with the following contents:

Word 1

S, 1-11 contains a two-character order for the 7631 File Control. These orders are described in detail in the reference manual, *IBM 1301 Disk Storage with IBM 7000 Series Data Processing Systems*. Form D22-6576-2. One of the following orders must be used:

82 – Prepare to verify, single record
83 – Prepare to write, format track
84 – Prepare to verify, track with no addresses
85 – Prepare to verify, cylinder operation
88 – Prepare to verify, track with addresses
89 – Prepare to verify, home address

21-35 contains the binary true address that is converted to BCD to perform the seek.

Word 2

S: If this bit is a 1, a write check is performed. If it is a 0, no write check is performed.

1: If this bit is a 1, special procedure is performed for no record found. If it is a 0, the standard procedure is performed. Both of these procedures are described under "Error Conditions," the next section.

2-35 is the BCD verification address, of the form TTTTRR.

The six-bit mode is used for all operations.

The channel command specified in the calling sequence by *ioparm,tl,* is used by IOOP as the data transmission command, not as the command to transmit orders for device control. Such commands are generated internally for the appropriate File Control orders.

ERROR CONDITIONS

The various error conditions that may be encountered in the use of disk storage, and the actions taken to correct them, are described below:

ERROR(S)	CORRECTIVE ACTION
Invalid Sequence Invalid Code Format Check Response Check Data Compare Check Parity Check Access Not Ready Circuit Check	The entire sequence is repeated up to four times. If the error persists, it is indicated to the calling program by an error return.
No Record Found Invalid Address	The entire sequence is repeated up to four times. After the fifth attempt to transmit data, a recalibration procedure is initiated. The sequence is then repeated four more times. If the error persists, it is indicated to the calling program by an error return.
No Record Found (Special Procedure)	The entire sequence is repeated up to four times. If the error persists, it is indicated to the calling

	program by an error return. (The use of this procedure may be specified only when disk storage is being used in random access fashion. See bit 1 of the second of the device orders specified by the calling sequence.)
Not Available Access Inoperative	Type operator message.

Upon the completion of a disk storage operation, the following bits are set in S.SSNS:

S.SSNS BIT	MEANING
3	Program Check
4	Data Check
5	Exceptional Condition
6	Invalid Sequence
7	Invalid Code
8	Format Check
9	No Record Found
10	Invalid Address
11	Response Check
12	Data Compare Check
13	Parity Check
14	Access Inoperative
15	Access Not Ready
16	1301 Circuit Check
17	7631 Circuit Check

1402 Card Read Punch

The maximum length of a physical record on a 1402 Card Read Punch is 80 BCD characters or 27 binary words. Bits 24-35 of word 27 are not transmitted when a record is written; they are set to zero when a record is read. An output block of more than 27 words is truncated to that length.

NON-DATA-TRANSMITTING OPERATIONS

The following non-data-transmitting operations may be used with the 1402 Card Read Punch:

1. Write a file mark on the punch. This operation causes the feeding of one card. If the operation is given for the card reader, the program error is indicated to the calling program by an error return.

2. Rewind the unit. This operation causes no activity.

3. Rewind and unload the unit. This operation causes an operator message to be typed.

ERROR CONDITIONS

The only 1402 error conditions that can be detected by IOOP are hole count, 1402 parity errors, and channel check. They are indicated to the calling program by an error return.

1403 Printer

Three models of 1403 can be connected to an IBM 7040/7044 Data Processing System. The 1403 Model 1 has a maximum print record size of 100 BCD characters and a speed of 600 lines per minute. The Model 2 has a maximum print record size of 132 BCD characters and a speed of 600 lines per minute. The Model 3 also has a print line 132 characters long, but its maximum speed is 1,100 lines per minute. If the user tries to write a print record longer than the maximum size allowed by the printer he is using, the record is truncated when the maximum number of words has been printed.

NON-DATA-TRANSMITTING OPERATIONS

The following non-data-transmitting operations may be used with the 1403 Printer:

1. Write a file mark. This operation causes the ejection of the page (skip to a punch in channel 1).

2. Rewind the unit. This operation causes no activity.

3. Rewind and unload the unit. This operation causes an operator message to be typed.

Before a record is written on a 1403 Printer, the first six bits of the first word of the record are examined. These six bits, which constitute the first character of the print line, are assumed to be a carriage control character intended for the printer. The following characters are translated into 1403 characters when they are encountered:

b (blank) or +	Normal single space before printing.
0	One extra space before printing (double spacing).
–	Two extra spaces before printing (triple spacing).

These characters are translated into the proper 1403 carriage control characters and are used with a control instruction to obtain properly spaced output.

The user may also use the actual 1403 carriage control characters, listed in Figure 10, in the first character position of the record.

If the first character of the print line is not a blank, plus sign, zero, or minus sign, it is assumed to be a valid 1403 carriage control character and is used unchecked and untranslated. It is the responsibility of the user to ensure that there is actually a valid character in this position.

After the carriage control character has been transmitted to the 1403, it is changed to a blank while the record is printed and is restored afterward.

ERROR CONDITIONS

The only 1403 error conditions that can be detected by IOOP are print timing, 1403 parity errors, and channel check. They are indicated to the calling program by an error return.

1622 Card Read Punch

The 1622 Card Read Punch reads and writes 80-character BCD records. If a request is made to IOOP to write

Control Character	Causes an Immediate Skip to	Control Character	Causes a Skip After Printing to
1	Channel 1	A	Channel 1
2	Channel 2	B	Channel 2
3	Channel 3	C	Channel 3
4	Channel 4	D	Channel 4
5	Channel 5	E	Channel 5
6	Channel 6	F	Channel 6
7	Channel 7	G	Channel 7
8	Channel 8	H	Channel 8
9	Channel 9	I	Channel 9
0 (not allowed)	Channel 10	? (12-0 punch)	Channel 10
= (8-3 punch)	Channel 11	. (12-8-3 punch)	Channel 11
' (8-4 punch)	Channel 12	) (12-8-4 punch)	Channel 12
	Immediate Space		Space after Printing
J	1 space	/ (0-1 punch)	1 space
K	2 spaces	S	2 spaces
L	3 spaces	T	3 spaces

Figure 10. 1403 Carriage Control Characters

a record more than 80 characters long, the record is truncated to 80 characters when it is written. A request for a binary operation on the 1622 results in a program error return.

NON-DATA-TRANSMITTING OPERATIONS

The following non-data-transmitting operations may be used with the 1622 Card Read Punch:

1. Write a file mark on the punch. This operation causes the feeding of one card. If the operation is given for the card reader, the program error is indicated to the program by an error return.

2. Rewind the unit. This operation causes no activity.

3. Rewind and unload the unit. This operation causes an operator message to be typed.

ERROR CONDITIONS

The possible 1622 errors, i.e., hole count and 1622 parity errors, cannot be detected by IOOP. If such an error occurs and a second request is made for an operation on the unit, a ready message is typed and IOOP enters a loop to await operator intervention. If a channel check occurs, the error is indicated to the calling program by an error return.

1009 Data Transmission Unit

The IBM 1009 Data Transmission Unit reads or writes messages that the IBM 1414 Input/Output Synchronizer separates into 80 character segments. The user's routine must reblock the input segments to obtain one message. The user's routine must make a separate entry to IOOP for each segment of an output message. If a request is made to IOOP to write a message segment more than 80 characters long, the segment is truncated. A request for a binary output operation on the 1009 results in a program error return.

NON-DATA-TRANSMITTING OPERATIONS (OUTPUT ONLY)

The following non-data-transmitting operations may be used with the 1009 Data Transmission Unit:

1. Write a file mark. This operation causes an operator message to be typed.

2. Rewind the unit. This operation causes an operator message to be typed.

3. Rewind and unload the unit. This operation causes an operator message to be typed.

ERROR CONDITIONS

The error conditions that may be encountered with the 1009 Data Transmission Unit are:

1. Input errors. The error is indicated by setting the message error flag in the input buffer (BPFLG).

2. Lost message. The error is indicated by setting the lost message flag in the input buffer.

3. Output transmission error. An error in transmission is indicated to the calling program by an error return. The user may transmit the message two additional times.

4. Output channel check. Two attempts are made to fill the buffer. If the condition persists, it is indicated to the calling program by an error return.

1011 Paper Tape Reader

The maximum length of a physical record on an IBM 1011 Paper Tape Reader is 80 BCD characters; bits 12-35 of word 14 are set to zero. A request for binary operations on the 1011 results in a program error return. It is the user's responsibility to ensure that the translation of 5-track or 8-track code to BCD representation is made by the 1011. The end of data must be recognized by the user, since the 1011 does not send an end-of-file signal. A subsequent call to read the device will result in a request to ready the unit.

The 1011 is considered to be a unit record device with calling sequences and error returns identical to those for a BCD card reader.

NON-DATA-TRANSMITTING OPERATIONS

The following non-data-transmitting operations may be used with the 1011.

1. Rewind the unit. This operation causes no activity.

2. Rewind and unload the unit. This operation causes an operator message to be typed.

ERROR CONDITIONS

The error conditions that may be encountered with the 1011 Paper Tape Reader are:

1. Input transmission. An error in transmission is indicated to the calling program by an error return.

2. Channel check. An error in data transfer from the 1414 to core storage is indicated to the calling program by an error return.

1014 Remote Inquiry Unit

The maximum length of a physical record on an IBM 1014 Remote Inquiry Unit is 80 BCD characters. If a request is made to IOOP to write a message more than 80 characters long, the message is truncated. The first character transmitted is the address (0 through 9) of the individual unit. A request for a binary output operation on the 1014 results in a program error return.

NON-DATA-TRANSMITTING OPERATIONS (OUTPUT ONLY)

The following non-data-transmitting operations may be used with the 1014 Remote Inquiry Unit:

1. Rewind the unit. This operation causes an operator message to be typed.

2. Rewind and unload the unit. This operation causes an operator message to be typed.

ERROR CONDITIONS

The error conditions that may be encountered with the 1014 Remote Inquiry Unit are:

1. Input errors. The error is indicated by setting the message error flag in the input buffer (BPFLG).

2. Output transmission error. An error in transmission is indicated to the calling program by an error return.

3. Output channel check. Two attempts are made to fill the buffer. If the condition persists, it is indicated to the calling program by an error return.

Telegraph Input/Output Unit

The telegraph input/output unit reads and writes 80 character BCD records. If a request is made to IOOP to write a message more than 80 characters long, the message is truncated. A request for a binary output operation on telegraph equipment results in a program error return.

NON-DATA-TRANSMITTING OPERATIONS (OUTPUT ONLY)

The following non-data-transmitting operations may be used with telegraph equipment:

1. Rewind the unit. This operation causes an operator message to be typed.

2. Rewind and unload the unit. This operation causes an operator message to be typed.

ERROR CONDITIONS

The error conditions that may be encountered with telegraph equipment are:

1. Input errors. The error is indicated by setting the message error flag in the input buffer (BPFLG).

2. Lost message. The error is indicated by setting the last message flag in the input buffer.

3. Output transmission error. An error in transmission is indicated to the calling program by an error return.

4. Output channel check. Two attempts are made to fill the buffer. If the condition persists, it is indicated to the calling program by an error return.

1401 Data Processing System

In conjunction with the 1401 Input/Output Control Program, IOOP uses the on-line 1401 as though it were an intermediate buffer between 7040/7044 core storage and the unit addressed. The limitations, non-data-transmitting operations, and error-recovery procedures for each device are essentially the same as those that exist when the device is attached to a 1414 Input/Output Synchronizer. The following devices can be read from or written on, using IOOP, when they are attached to an on-line 1401:

1. Up to six 729 or 7330 Magnetic Tape Units (maximum record length: 370 words)
2. A 1402 Card Read Punch
3. A 1403 Printer

The Input/Output Executor (IOEX) consists of a channel scheduler, a program scheduler, a CPU trap supervisor, and a collection of utility routines.

The channel scheduler controls the order in which input/output operations are performed on each of the channels.

The program scheduler determines the order in which programs and routines are executed by the computer. It can be used in conjunction with the channel scheduler to perform real-time or random processing.

The CPU trap supervisor takes action on the various kinds of traps that are not caused by input/output operations controlled by the channel scheduler. These include invalid instruction traps, interval timer traps, storage protection violation traps, and storage parity error traps.

Utility routines provide four kinds of conversion from binary to BCD or octal and the facilities necessary for writing on the console typewriter.

Applications of IOEX

Programming of input/output at the IOEX level is recommended only for the most unusual kinds of input/output operations. Under normal circumstances, only operations on devices that are not supported by IOCS (and real-time applications) justify direct use of IOEX. Most of the facilities (including the scheduling of special routines) that are avaialble at the IOEX level are also available at the IOOP level.

Use of IOEX

Figure 11 shows the sequence of routines involved in the execution of an input/output operation. Although the select and error-recovery routines are shown as user-written, this may not always be the case. The user may wish to supplement, rather than to replace the standard select and error-recovery routines supplied with IOOP.

For the purposes of this discussion, the following terminology is defined:

Activity

Use of a device that makes both the device and the channel busy. An activity is internally initiated by a select, sense, or control instruction, and is terminated by an end-of-operation trap.

Event

Use of a device that makes the device busy, but does not make the channel busy. An event commences at an end-of-operation trap, and is terminated by an attention trap.

Dormancy

A special type of event peculiar to real-time input devices. It comprises a passive waiting period of indefinite duration and a subsequent "active" period that is not internally initiated. The event, therefore, is not scheduled, but is initiated by some external action, e.g., an attention from an IBM 1014 Remote Inquiry Unit. The attention that signals the termination of the "active" period (and the state of dormancy) is thus not predictable within any given period of time and may not occur. Note that a dormant device presents to the computer the appearance of being busy, i.e., it cannot be read.

Operation

A sequence of events and activities on a particular device, such as writing and error correcting a record on magnetic tape, punching a card, backspacing several records on magnetic tape, or seeking, writing, and write checking a track on disk storage. An operation is initiated and terminated by a series of entries to a select and error-recovery routine.

The following sections describe the functions of IOEX and the ways in which it can be used by the programmer.

Unit Control Blocks

During the assembly of the Nucleus, one nine-word unit control block is generated for each input/output device connected to the computer. For the purposes of this discussion, each unit connected to a 1414 Input/Output Synchronizer, a 7631 File Control, or directly to the 7040/7044 is considered to be a device.

The unit control blocks are generated in order of priority of device, and the unit control blocks for each channel form a continuous string. When IOEX is ready to initiate a new operation on a given channel, it scans over the unit control blocks of that channel, from high-priority devices to low-priority devices, searching for a waiting operation. The structure of a unit control block is shown in Figure 12. The significance of the two-character bit codes is:

EI

= 0 if this device is internally initiated (i.e., can be read and/or written at will).

= 1 if this device is externally initiated (i.e., signals when it is ready to be read or written).

DA

= 0 if this device is assigned to some symbolic unit.

= 1 if this device is not assigned to any symbolic unit (i.e., is not associated with any system control block).

RW*

= 0 if a write or non-data-transmitting operation (other than forward space) is in progress or has been requested.

= 1 if a read or forward space operation is in progress or has been requested.

ER*

= 0 if no event has been requested.

= 1 if an event has been requested.

EC**

= 0 if no event is in progress.

= 1 if an event is in progress.

RB – reserved for future use by DS Programming Systems.

IP

= 0 if an external interrupt has occurred.

= 1 if the device is dormant.

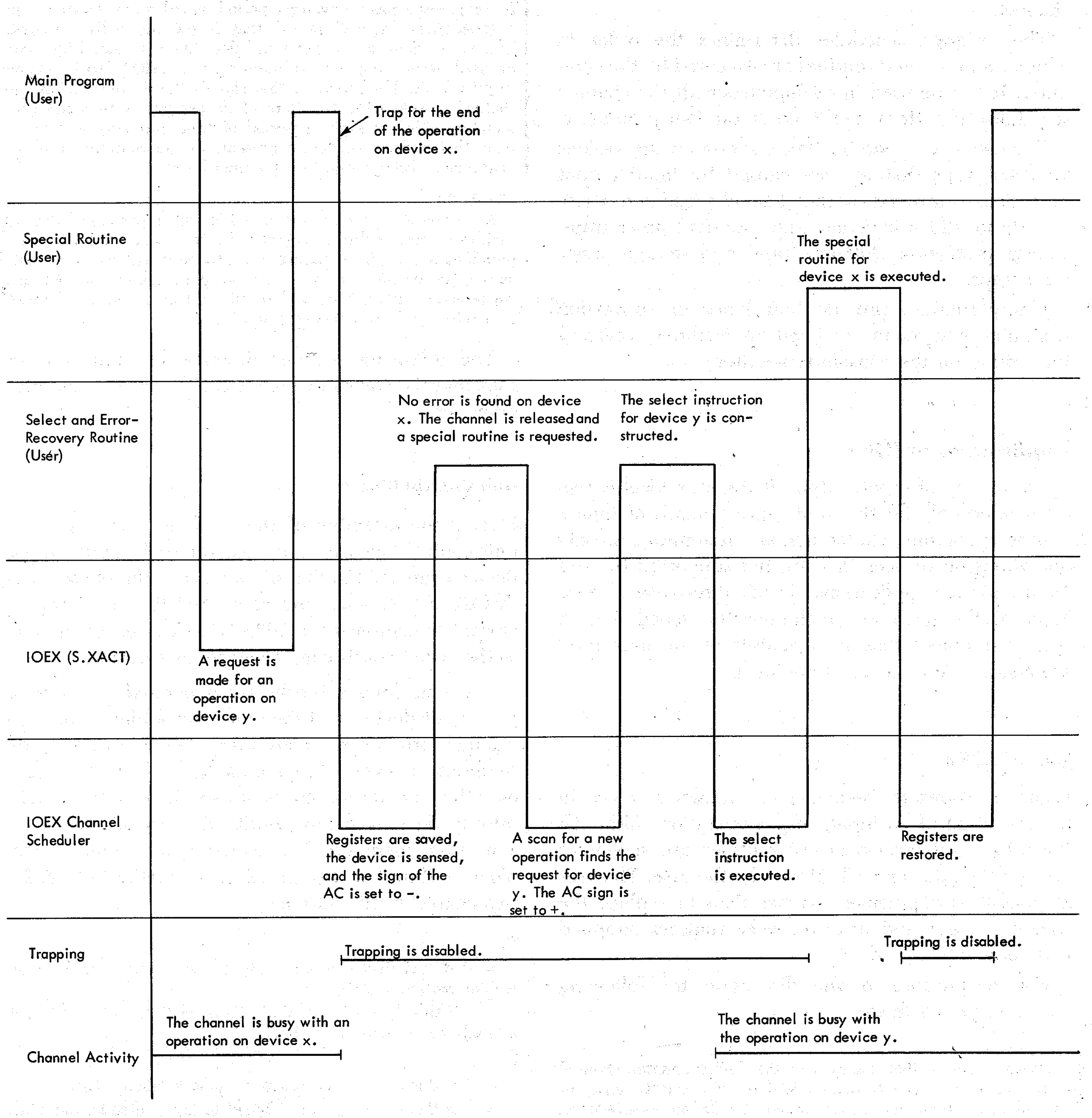
PR**

= the priority of the special routine. Priorities from 0 to 4 may be used by the calling program.

MP**

= 1 if a device-not-ready message has already been printed for this operation.

*Inserted by S.XACT.
**Maintained by select and error-recovery routines.



Trap for the end of the operation on device x.

The special routine for device x is executed.

No error is found on device x. The channel is released and a special routine is requested.

The select instruction for device y is constructed.

A request is made for an operation on device y.

Registers are saved, the device is sensed, and the sign of the AC is set to –.

A scan for a new operation finds the request for device y. The AC sign is set to +.

The select instruction is executed.

Registers are restored.

Trapping is disabled.

Trapping is disabled.

The channel is busy with an operation on device x.

The channel is busy with the operation on device y.

Figure 11. Simplified Diagram of the Functioning of IOEX

	S	1	2	3 – 14	15 – 17	18 – 20	21 – 35	
1. UCUNI	E I	D A		Device sub-address	Interface	▨	Device select address for BCD operation	
2. UCPOS	Information that defines the position of the device **							
3. UCSEL	R W	E R	E C	Location of the select and error-recovery routine used for this device *		▨	Effective address of the pointer to this unit control block *	
4. UCPRC	R B	▨	▨	Number of permanent read errors on this device **		▨	Number of recovered read errors on this device **	UCRRC
5. UCCRC	M P	▨	▨	Number of consecutive attempts to recover an error **		▨	Number of recovered write errors on this device **	UCWRC
6. UCRXC	▨	▨	▨	Number of exception records read on this device **		▨	Number of records read on this device **	UCRDC
7. UCWXC	▨	▨	▨	Number of exception records written on this device **		▨	Number of records written on this device **	UCWDC
8. UCSTG	I P	▨	▨	Code indicating the stage that has been reached by the current operation on this device **		PR	Location of the entry point to a special processing routine for this operation **	UCRTN
9. UCSEN	Sense data for this device			Reel Serial Number (used by IOLS)				UCCSN

* Inserted by S.XACT.
** Maintained by select and error-recovery routines.

Figure 12. Format of the Unit Control Block

System Data Locations

In addition to the unit control blocks, IOEX uses a series of data words to keep track of input/output operations. These locations, which may be referred to by the user's select and error-recovery routines, are:

S.XTDT
contains the trap condition indicators.

S.XLTP
indicates the position of the device before the trap occurred.

S.XSCH
contains the information that was in the channel register after the trap occurred.

S.XCPS
is negative if a checkpoint is being taken.

S.XTPS
is negative if trapping is inhibited.

S.XSNS and S.XSNS+1
contain the sense data obtained by IOEX before each entry to a select plus (activity/event initiation) routine, and before each entry to the select minus (activity/event checking) routine for an activity or event that caused an unusual-end trap.

Requests for Operations

The user can request the use of a channel for an operation on a given device with the following calling sequence:

```
TSX      S.XACT, 4
pfx      p(ucb), t, selrtn
```

where:

pfx
S bit = 0 for a write or non-data-transmitting operation (other than forward space).
= 1 for a read or forward space operation.
1 bit = 0 if no event is requested.
= 1 if an event is requested.
2 bit = 0 for a normal request.
= 1 for a priority request.

p(ucb), t
is the location of a pointer to the unit control block of the device on which the operation is to be performed. This pointer has the form:

S	2 3 – 17	18 – 20	21 – 35
			l(ucb)

where:

l(ucb)
is the location of the first word of the unit control block. This word may be an entry in the Symbolic Units Table (the prefix, decrement, and tag portions are not used by IOEX).

selrtn
is the location of the select and error-recovery routine that is to be used to initiate and check this operation.

The S.XACT routine places bits S and 1 of *pfx*, and *selrtn*, in the prefix and decrement positions of word

UCSEL of the unit control block specified. The fact that the decrement of UCSEL is nonzero indicates to IOEX that an operation on this device has been requested. Similarly, it should be used by the programmer to determine whether or not an operation on the device can be requested. The S.XACT routine does not check to see if an operation is already waiting or in progress, and there is no facility available for stacking requests. If a request is made for an operation on a unit that already has an operation waiting or in progress, this previous operation is deleted. Unless he intends to delete the waiting operation, the user should wait until the decrement of UCSEL becomes zero (indicating the release of the unit by the previous select and error-recovery routine) before entering S.XACT with a new operation.

To allow input activity on a real-time (e.g., telecommunications *input*) device, the user issues a standard calling sequence to S.XACT, supplying the address of his select and error-recovery routine. If desired, the standard select routine contained in the IOOP2 level of IOCS may be specified (this address is found in word SCSEL of the system control block for this device). This call prepares IOEX to accept attention signals from the device as requests for operations. When an attention occurs, the user's select routine is entered. The select-minus portion of this routine should not clear the request posted in word UCSEL of the unit control block by the initial call to S.XACT, but should set the device dormant (IP bit of UCSTG = 1). To discontinue real-time input activity, the user can call S.XACT, omitting field *selrtn* of the calling sequence.

To request output operations on a real-time device, a call to S.XACT must be executed for each message, i.e., buffer, to be written.

CHANNEL PRIORITIES

If the user requires immediate use of a channel for an input/output operation, he can make a priority call to IOEX. This is done, as was stated earlier, by making the 2 bit of *pfx* a 1. As each activity required by an operation is completed, IOEX gives control to the select and error-recovery routine to allow checking of the activity. After control is returned to IOEX by this routine and before the routine is re-entered to initiate the next stage of the operation, IOEX checks to see if a priority operation has been requested on the channel. If this is the case, control is passed to the select and error-recovery routine for the priority operation immediately after all events on the channel have been initiated, so that this operation can be initiated.

The IOEX allows for only one priority request at a time on a given channel. If a priority request is made before a previous priority operation has been completed, the first priority operation is interrupted in the same way that a normal operation would have been. Consequently, it is urged that priority requests to IOEX be used for real-time applications only, so that a normal program does not interfere with a real-time program that may be operating at the same time.

If an attention is signaled by an externally initiated device, the device is given priority on the channel. Any previously requested priority on the channel, either by a prior attention or by a priority call to S.XACT, is canceled.

DATA CHANNEL TRAPS

IOEX enables the computer to recognize the following conditions as causes of data channel traps:

1. Completion of an activity (end-operation trap)
2. Redundancy check while writing
3. Reading of a file mark
4. Word parity check
5. Unusual-end signal from an adapter
6. Unit-record attention
7. Attention from an adapter

The first six types are termed *internally initiated* traps, because they are caused by currently scheduled activities and because they are expected to occur when the channel ceases to be busy. The seventh condition may be internally initiated, if it is the termination of an event such as a seek on a disk storage unit, a read or write on a unit record device, or a write on a telecommunications device. The seventh condition may be *externally initiated*, if its cause is the termination of dormancy. This includes an input message waiting on a telecommunications device.

When an attention trap occurs, IOEX determines the cause of the trap. If it was caused by the termination of an event, the event-in-progress (EC) bit is reset in the unit control block and, if the channel is busy, control is returned to the program that was interrupted, internally initiated interrupts are processed when the current channel operation has been completed.

If the trap was caused by the termination of dormancy, the external interrupt (IP) bit is reset in the unit control block, the unit is given channel priority and, if the channel is busy, control is returned to the program that was interrupted. Externally initiated interrupts are processed when the current channel activity has been completed.

SELECT AND ERROR-RECOVERY ROUTINES

The programmer using the IOEX level of IOCS must provide a select and error-recovery routine for each kind of device he will use. The select and error-recovery routine constructs the select instructions, channel commands, and device orders necessary for each event and activity in a given operation; keeps track of the stage that the operation has reached; and examines the results of each trap for errors. If a given activity or event is unsuccessful, the select and error-recovery routine must initiate an error-recovery procedure appropriate to the device.

Whenever IOEX enters a select and error-recovery routine, storage protection has been released, traps are inhibited, index register 1 contains the channel number, index register 4 contains the calling linkage, and the address portion of the accumulator contains the address of the unit control block. If the programmer wishes to use index registers 1 or 4, he must save their contents and restore them before returning control to IOEX.

When the select and error-recovery routine is entered, the sign of the accumulator serves as a switch, indicating the kind of processing required. If the sign is plus, it indicates that the channel is available for the initiation of events and activities on the specified device (either the beginning of a new operation or the continuation of an existing one). If the sign of the accumulator is minus, the select and error-recovery routine is to check a completed activity or event for errors.

The select and error-recovery routine is divided into two sections. One is called *select plus* and the other is called *select minus*. The considerations affecting each of them are discussed separately.

SELECT PLUS

When the select plus routine is entered, IOEX has left in locations S.XSNS and S.XSNS+1 the results of a sense instruction on the device to be treated. Select plus must determine whether or not the device is ready. If it is not, or if, for some other reason, the select plus routine is not ready to issue a select instruction for the device, control should be returned to IOEX by a TRA 2,4. If the select plus routine specified for a priority operation returns by a TRA 2,4, the operation loses its priority. The entire request can be deleted by zeroing out the prefix and decrement of word UCSEL in the unit control block before returning. When a return is made to IOEX by a TRA 2,4, it enters the select plus routine for another device and schedules the first operation later.

If the device is ready, the select plus routine must construct the select instruction, channel command, and device order required by the current stage of the operation. It then returns control to IOEX by a TRA 1,4; leaving the select instruction in the accumulator and the channel command in the MQ. For a non-data-transmitting operation, which requires no channel command and hence no Reset and Load Channel instruction, the MQ should be cleared. The IOEX executes the select instruction and then (if necessary) a Reset and Load Channel instruction for the channel command.

Certain stages in an operation, such as the sequence of "Prepare to Verify" and then "Write Select" on a 1301 Disk Storage unit, have timing considerations that make separate scheduling of each activity in the sequence inefficient. Both of these activities should be considered as one stage of the operation, and IOEX allows for multiple entries to the select plus routine to allow them to be performed together.

If the sign of the MQ is minus when the select routine returns control to IOEX, the necessary set of instructions is executed immediately and select plus is re-entered for another set. On the subsequent entries, the sign of the accumulator is not set to plus, nor is the location of the unit control block placed in the address portion of the accumulator. For devices that require such a multiple entry, the select and error-recovery routine might begin with a BRA/BRN switch. This switch would be set to BRA before control is returned to IOEX to have the first activity initiated and reset to BRN after the final entry.

When trapping is enabled after a multiple entry to select plus, S.XTDT, S.XSNS, and S.XSNS+1 contain a logical OR of the results of all activities and events initiated by the multiple entry.

SELECT MINUS

If the sign of the accumulator is minus when control is given to the select and error-recovery routine, an end-of-activity trap has occurred. The select minus routine must examine the results of the trap (in location S.XTDT) and the contents of the channel register (in location S.XSCH) to determine whether or not the activity was successful. Entry to select minus after an event is conditional unless an unusual end has occurred. If an unusual end has occurred, sense data is to be found in location S.XSNS. Channel parity errors are handled by the storage parity routine and are not communicated to the select minus routine.

If an error is detected or if the operation requires additional activities or events for its completion, select minus should record information in the decrement of word UCSTG in the unit control block that will inform select plus of the stage that the operation has reached. If an error occurred during a multiple entry to select plus, all activities should be considered as one stage and the entire sequence should be repeated.

Select minus routines must maintain the following portions of the unit control block:

1. The number of consecutive errors, in the decrement portion of word UCCRC

2. The number of exception (e.g., noise) records, in the decrement portion of word UCRXC or word UCWXC (reading or writing)

3. The number of permanent read errors, in the decrement portion of word UCPRC

4. The number of recovered errors, in the address portion of word UCRRC or word UCWRC (reading or writing)

5. The number of records read or written, in the address portion of word UCRDC or word UCWDC

6. The position of the unit, in word UCPOS

The position of a sequential device, as recorded in word UCPOS of the unit control block, consists of the number of file marks written or spaced to, in the decrement of the word, and the number of records written or read since the last file mark, in the address portion of the word.

The device position is defined separately for each type of random access device. For 1301 Disk Storage, the position of the unit is given by the current head and track number, in binary, in the address portion of word UCPOS.

As long as more activities or events (i.e., more entries to select plus) are required to complete the operation, select minus should return to IOEX by a TRA 1,4. If the activity is completed at an end-of-operation trap but an event is still in progress, select minus sets a 1 in the EC bit of word UCSEL, indicating that an event is in progress. IOEX zeroes the EC bit of UCSEL when the event is completed. When no additional events or activities are required, select minus must clear the prefix and decrement portions of word UCSEL, freeing the device for another request, and return control to IOEX by a TRA 2,4.

The select minus portion of the select and error-recovery routine for a real-time input device must set the IP bit in the unit control block after a completed activity. In no case should select minus clear the prefix and decrement portions of word UCSEL for this type of device.

SPECIAL ROUTINE REQUESTS

Before returning to IOEX at the end of an operation, select minus can request the execution of a special routine. This is done by placing the location of the entry point to the special routine in the address portion of word UCRTN, and the priority of the routine (a number from zero through four) in the tag portion of the word. Control is then returned to IOEX by a TRA 2,4. For a discussion of priorities and scheduling of special routines, see the section "Program Scheduling."

USE OF STANDARD SELECT AND ERROR-RECOVERY ROUTINES

The user can write routines that assign buffer areas, schedule (queue) a particular activity on the unit, provide additional error procedures, and then transfer control to the standard IOCS select and error-recovery routine for the device. The location of the entry point to the standard select and error-recovery routine is the address portion of the word SCSEL in the system control block. This word is described in the section "System Control Block" in the chapter "Input/Output Operations (IOOP)."

On the select plus entry to the user's routine, if a TRA 2,4 return to IOEX is not to be taken, the user must provide the information in locations SCOPR, SCIOC, PR, SCRTN, SCD01, and SCD02. If the decrement portion of word UCSTG is not zero, the operation has begun and the intermediate stages are probably of no interest. The contents of index registers 1 and 4 and of the algebraic accumulator must be preserved for transfer to the standard select and error-recovery routine.

On the select minus entry to the user's routine, the user may save index register 4, and TSX to the standard select and error-recovery routine. Index register 1 and the accumulator must not be disturbed. If the standard select and error-recovery routine returns to 1,4, the operation is not complete. The user's routine can reload index register 4 and return to IOEX by a TRA 1,4. If the standard select and error-recovery routine returns to 2,4, the operation is complete, error counts and device position have been updated, and the special routine request (if any) is in word UCRTN. Index register 1 must be preserved. By examining the contents of word SCSCH in the system control block, the user's routine can determine if the operation was completed successfully or if a permanent error exists. The user may post completion indicators and he may update queues. If additional operations on the device are required, the user must restore the location of the entry point to his select routine in the decrement portion of word UCSEL, which was cleared by the standard select and error-recovery routine when the device was released.

Select routines operate with storage protection released. They should not be debugged during system runs. Select routines should be short, so that real-time processing is not prevented while data channel traps are inhibited.

Channel Scheduling

The IOEX is responsible for scheduling the use of the data channels by the devices attached to them. It assigns priorities for a given channel in the following order:

1. Entry to the select and error-recovery routine at the end of a current activity.
2. Events to be initiated.
3. Priority requests to S.XACT or externally initiated input activities.
4. More activity on the current operation.
5. A new operation.

When the computer is trapped, the following occurs:

1. The select minus entry on the current activity is always serviced first. Select minus may return to IOEX with information on the stage reached by the operation. Select minus can also return to IOEX with a special routine request. Entry to select plus to initiate the next stage of the operation and entry to the special routine may be delayed because of scheduling considerations.
2. If any events have been requested, IOEX enters select routines for all events to be initiated on the channel.
3. If a priority request exists, it takes the place of any incomplete currently scheduled operation. Thus, a currently scheduled operation that is not completed may be temporarily discontinued.
4. The currently scheduled select plus entry is made. This will be a continuation of an incomplete operation only if no priority request has superseded it.
5. If there is no currently scheduled operation, the unit control blocks for the channel are scanned for a new operation. This new operation may be, in fact, a resumption of a delayed incomplete operation.
6. If no new operation is found, the channel is permitted to become idle.

If one or more operations on the channel requests rescheduling by a 2,4 return from select plus, IOEX goes into a hold loop until the channel can be reactivated. The loop is broken for any of the following conditions:

a. Any other channel busy
b. An event in progress on any channel
c. An attention from an externally initiated device

When the hold loop is entered, the following action will be taken for a given device:

Contents				Enter Select Plus	Release Hold Loop	Remarks
EI	EC	IP	UCSEL			
0	0	0	0	no	no	attention, but no request
0	0	0	sel	yes	yes	request, event completed
0	1	0	0	no	yes	no request, event in progress
0	1	0	sel	no	yes	request, event in progress
1	0	0	0	no	no	attention, but no request
1	0	0	sel	yes	yes	request, device not dormant
1	0	1	0	no	no	no request, device dormant
1	0	1	sel	no	no	request, but device dormant

Program Scheduler

The program scheduler is the portion of IOEX that determines the order in which non-trap-time programs are executed by the central processing unit.

When an operation is completed by a given trap, the select minus routine can request that a special routine be executed. When IOEX receives the TRA 2,4 return, which indicates the end of the operation, and after it has initiated a new operation on the channel (if possible), it transfers control to the program scheduler.

Program Priorities

The program scheduler determines whether or not a routine of higher priority than the program in progress has been requested by the select minus routine. If so, the contents of s.SSCH, s.SSNS, the location interrupted, and the register contents saved by IOEX when the trap occurred are all stored in a push-down list. Trapping is re-enabled, and the new routine is entered by a TSX sprtn,4.

Routines may be assigned any priority from 0 through 4. The main program has no priority. Priorities 5 through 7 are reserved for the system. The length of the push-down list, which limits the number of levels of priority that can be used effectively, is an assembly parameter in the symbolic input for the System Monitor. If a routine of higher priority than the program in progress is requested, but the push-down list is full, the higher priority routine is not entered.

When a routine returns to IOEX, the unit control blocks are scanned for the highest priority routine waiting to be entered. If none of these has a higher priority than the latest program placed in the push-down list, this program is re-entered at the point at which it was interrupted.

Use of Special Routines

When a special routine is entered by IOEX, index register 4 contains the calling linkage and the address portion of the accumulator contains the location of the first word of the unit control block in which the special routine was requested. The address and tag portions of word UCRTN of this unit control block have been cleared by IOEX.

A special routine may use all levels of IOCS; however, the IOBS and IOLS levels should not be used if there is the possibility that either of them is in use at the time that the special routine is executed. A special routine can request scheduling of another special routine (but not by placing its address in a unit control block, because storage protection has been turned on). Since trapping is enabled before the special routine is entered, it is possible that the special routine may be interrupted by another routine of higher priority. For this reason, some care should be taken in using information from the unit control block through which the special routine was called, as another special routine may already have requested an operation on the unit and changed the unit control block.

When the special routine has completed all processing, it must return control to IOEX by a TRA 1,4.

Deactivation of IOEX

The symbol S.XDAC is the entry point to an IOEX subroutine that deactivates all channels for the purpose of taking a checkpoint or dump. This subroutine cannot be called by a nonstandard select routine.

The calling sequence to this routine is:

```
          TSX       S.XDAC,4
          pfx       ptr
```

where:

pfx
= MZE to deactivate all channels and special-routine requests.
= PZE to nullify a previous deactivation and to activate all possible channels.

ptr
= the location of a pointer to a unit control block of the form
    pfx   l(ucb),b,c
in which pfx, b, and c are ignored
(ptr may be a word in the Symbolic Units Table).

If ptr exists, IOEX will permit channel activity for this unit only.

The effect of an MZE call to S.XDAC is:

1. A switch is set to prevent any call to a select-plus routine except for externally initiated input devices, for any operation on the unit specified in ptr, and for any activity resulting in an event.

2. A switch is set to prevent any entry to a special routine.

3. Each channel is tested for a lost trap. If a trap was lost, the channel is set inactive, and UCSEL cleared for the device.

4. If an activity was requested on the device corresponding to unit ptr and the channel is not active, the channel is activated.

5. A switch is set to prevent holding in IOEX for any unit except for the device corresponding to unit ptr.

The effect of a PZE call to S.XDAC is:

1. The switches mentioned above are reset.
2. All channels are activated.
3. A waiting special routine is entered.

A call to S.XDAC has no effect on calls to select minus for end-of-operation or unusual-end traps.

Central Processing Unit Traps

The CPU trap supervisor processes traps that are not the result of input/output operations. The following actions are taken for each variety of trap:

1. *Interval Timer Reset:* The job is terminated with a dump.

2. *Storage Protect Violation:* The job is terminated with a dump.

3. *Storage Parity Error:* The job is terminated with a dump, unless parity trapping was inhibited when the error occurred.

4. *Instruction Traps:*
   a. Store Location and Trap* – the job is terminated with a dump.
   b. Floating Point* – the trap is ignored.
   c. Release Protect Mode – if this is a monitor violation, the job is terminated with a dump.
   d. Set Protect Mode – the job is terminated with a dump.

5. *Interval Timer Overflow*:* The job is terminated with a dump.

6. *Direct Data Trap*:* The trap is ignored.

The processing of the conditions marked * can be changed by substitution of a user-written routine for the IOCS routine. The program can modify the trap transfer location for the particular error by a call to S.SCCR. The System Monitor restores the standard trap transfer each time it is loaded. The return of control to the interrupted program is the responsibility of the user's routine. No return to IOEX should be made.

Utility Routines

The IOEX contains several utility routines for the convenience of the programmer. These routines include several conversion routines and a general typewriter routine for operator action messages.

Conversion Routines

Four basic conversion routines are provided within IOEX. They are:

1. Binary to octal conversion of the data in the address portion of the accumulator. The call is:

```
          TSX       S.XOVA,4
          Return
```

The result is right-justified in the accumulator.

2. Binary to octal conversion of the data in the decrement portion of the accumulator. The call is:

```
          TSX       S.XOVD,4
          Return
```

The result is right-justified in the accumulator.

3. Binary to decimal conversion of the data in the address portion of the accumulator. The call is:

```
          TSX       S. XDVA, 4
          Return
```

The result is right-justified in the MQ.

4. Binary to decimal conversion of the data in the decrement portion of the accumulator. The call is:

```
          TSX       S. XDVD, 4
          Return
```

The result is right-justified in the MQ.

Typewriter Routine

The calling sequence to the typewriter print routine is:

```
          TSX       S. XPRT, 4
          PZE       fword, , count
```

where:

fword
  is the location of the first word of the message to be typed.

count
  is the number of BCD characters to be typed. This count is converted to words before the select instruction is executed.

The typewriter routine ignores errors.

Input/Output Label System (IOLS)

The use of labels is one of the most inexpensive ways to prevent avoidable failures of jobs. Label verification ensures that the correct files are read, in the right sequence, and that valuable information is not destroyed. Labeling also allows the programmer to treat his files in a more general manner than would be possible if he had to read or write each reel of magnetic tape or each section of disk storage separately, and it reduces the amount of internal housekeeping by providing indications of the ends of files and of segments of files.

The Input/Output Label System (IOLS) performs all of the label verification and creation functions required for opening and closing, reel switching, and forcing the end-of-reel condition on both labeled and unlabeled files. It can be used with either the IOOP level or the IOEX level of input/output programming, but it requires the presence in core storage of IOOP for reading and writing labels and for positioning devices.

Format of Labeled Files

If IOLS is to be used on a given file, this file must be organized in the following manner:

1. Beginning of medium indicator (or the end of the previous file)
2. Header label, in BCD mode
3. File mark
4. Checkpoint, if any, in binary mode
5. File mark, if there is a checkpoint
6. Data blocks, in either BCD or binary mode
7. File mark
8. End-of-reel or end-of-file trailer label, in BCD mode
9. File mark

In multi-reel files, item 8 is an end-of-reel trailer label for each reel except the last one, on which it is an end-of-file trailer. If a given unit contains more than one file; items 2 through 9 are repeated for each file.

Format of Labels

The IOLS verification and creation routines are designed to operate with the IBM standard label. The format of this label is shown in Figure 13. The standard label is 120 characters (twenty 7040 words) long. It is possible that the programmer may prefer to create and verify his own labels, using IOLS only for reading and writing labels and for unit switching. In this case, IOLS performs no verification or creation of its own. Only labels exactly 120 characters long are written, however, and any label longer than 120 characters is truncated to that length when it is being read into core storage.

Label Control Block

In the process of checking labels on input files and creating labels for output files, IOLS uses the labeling information supplied by the programmer for each file. This information is held in a nine-word label control block. When IOLS is being used as a part of IOBS, the label control block is part of the file control block (words 8-16). The format of the label control block is shown in Figure 14.

The explanation of the two-character bit codes in the label control block are:

OB—Open Status Bit
= 0 if the file is not open.
= 1 if the file is open.

R1, R2 – These are reserved for future use by DS Programming Systems.

PR – Permissible to Reposition
= 0 permissible to Reposition at CLOSE
= 1 do not reposition at CLOSE

NA*– Nonstandard or Additional Label Processing
= 0 if the label is a standard IBM label and is to be verified by IOCS. If the address in bits 21-35 of this word is not zero, this address is the location of the entry point to an additional processing routine.
= 1 if the label is not a standard IBM label and is not to be verified by IOCS. The label is read into and written from the label area. If the address in bits 21-35 of this word is not zero, this address is the location of the entry point to a nonstandard verification or creation routine.

IN—Input
= 0 if the file is output.
= 1 if the file is input.

CI*– Checkpoint Indicator
= 0 if there are no checkpoints on this file.
= 1 on an input file, if a checkpoint follows the header label on each reel of the file and is to be automatically passed over by IOLS.
= 1 on an output file, if a checkpoint is to be written following every header label.

MI—Mode Indicator
= 0 if the file is in BCD mode.
= 1 if the file is in binary mode.

RS*– Reel Specification
= 0 if the file is single-reel.
= 1 if the file is multi-reel.

LI*– Labels Indicator
= 0 if the file is unlabeled.
= 1 if the file is labeled.

SC*– Sequence Check
= 0 if no block sequence numbers are formed or checked for this file.
= 1 if block sequence numbers are formed or checked for this file.

CC*– Check Sum Check
= 0 if no check sums are formed or checked for this file.
= 1 if a check sum is formed or checked for each record written or read on the file.

Field	Positions	Field Name	Description
1	1-5	Label Identifier*	1HDRb to indicate a header label, 1EORb to indicate an end-of-reel trailer label, or 1EOFb to indicate an end-of-file trailer label.
	6		Blank.
2	7-10	Retention Period*	The number of days after the creation date (0000-9999) that this file is to be retained.
3	11-15	Creation Date*	The year and day of the year on which the file was created. The year occupies the first two positions (00-99) and the day of the year occupies the last three positions; e.g., December 30, 1963 would be entered as 63364.
4	16-25	File Identification*	A ten-character field identifying the file.
5	26-30	File Serial Number*	This field is the same as the Reel Serial Number of the first or only reel of the file.
6	31-35	Reel Serial Number	A five-character identification code assigned to the reel when it enters the installation. This number normally appears on the outer surface of the reel for visual identification.
	36		Blank.
7	37-40	Reel Sequence Number*	A four-digit number (0001-9999) that is the order of this reel within the file.
	41		Blank.
8	42-44	Reserved	This field is reserved for future DS Programming Systems use.
9	45	Density Indicator***	This field specifies the density change that is necessary before the file is read: 0 — The body of the file is in the same density as the label. The 7040 cannot process files written in varying densities.
10	46	Check Sum Indicator**	This character indicates the presence (1) or absence (0) of check sums.
11	47	Block Sequence Indicator**	This character indicates the presence (1) or absence (0) of block sequencing.
12	48	Recording Mode Indicator**	This character indicates whether the file is in BCD (2) or binary (1) mode.
13	49	Recording Technique Indicator***	This field specifies the number of bits recorded as one byte: 6.
14	50	Data Processing Technique Indicator***	This field specifies the number of bits to be processed as one byte: 6.
15	51-54	Creating System	This field specifies the system that created the file. Labels created by IOLS always have the characters 7040 in this field.
16	55	Record Format***	This character indicates the record format found in this file.
17	56-60	Record Length***	For fixed-length records, this field specifies the number of characters in each data record; for variable-length records, it specifies the number of characters in the largest possible data record in the file.
18	61-65	Block Size***	For fixed-length records, this field specifies the number of data records in each block; for variable-length records, it specifies the number of characters in the largest possible block in the file.
19	66	Checkpoint Indicator**	This character indicates the presence (1) or absence (0) of checkpoints in the file.
20	67-72	Block Count*	This field specifies the number of blocks written on this reel of the file (excluding labels, checkpoints, and file marks). The block count is given in trailer labels only.
21-26	73-100	Reserved	These fields are reserved for future DS Programming Systems use.
27	101-120	For Optional Use***	Positions 101 through 120 may be employed as an area for additional label data, at the option of the user.

*This field is verified by IOLS.

**This field overrides and replaces the corresponding label control block specifications.

***This field is not interpreted by IOLS.

Figure 13. Format of the Standard Label

	S	1	2	3	4 5 6	7	8	9	10	11	12 13 14 15 16 17	18	19	20	21 22 23 24 25 26 27 28 29 30 31 32 33 34 35
1. LCUNI	O B	R 1	▨	Address of Symbolic Units Table entry of the secondary unit *								P R	R 2	▨	Address of Symbolic Units Table entry of the primary unit *
2. LCFCN	N A	▨	▨	Reserved								▨	▨	▨	Address of a nonstandard or additional label processing routine *
3. LCCON	I N	▨	▨	C I	▨	M I	R S	L I	S C	C C	▨	▨	▨	▨	Available to the user
4. LCBSN	▨	▨	▨	Available to the user								▨	▨	▨	Number of blocks read or written on this unit *
5. LCFSN	File serial number, in BCD, of the form 0XXXXX *														
6. LCRSN	Reel sequence number, in BCD, of the form 00XXXX *														
7. LCRET	For an output file, the retention period (days) in BCD, of the form 00XXXX * For an input file, the file creation date in BCD, of the form 0XXXXX *														
8. LCID1	First six characters of the BCD file identification, of the form XXXXXX *														
9. LCID2	Last four characters of the BCD file identification, of the form XXXX00 *														

Fields marked with an * must be set before any IOLS routine is used.

Figure 14. Format of the Label Control Block

Fields marked with an * must be set before any IOLS routine is used. If the decrement of word LCUNI is zero, the primary unit is used as the secondary unit. If S.SIN1, S.SOU1, or S.SPP1 is the primary unit, the secondary unit field should be either zero or the same as the primary unit field. Unit switching on these three units is always completely automatic.

Calling Sequence

The same general calling sequence is used to call all four IOLS routines. It is written as:

```
        TSX      S.IOLS, 4
        oper     lcb, t, label
        pfx      erret, , mode
        ret1
        ret2
```

where:

oper
is one of the following four operation codes:

ILOPN	Open the file.
ILRSW	Switch reels.
ILFER	Force the end-of-reel condition.
ILCLS	Close the file.

The actions taken by IOLS for each kind of operation are detailed later.

lcb, t
is the location of the label control block for the file.

label
is the location of a 120-character buffer provided by the programmer for the use of IOLS. Any labels that must be read into core storage for this operation, and all labels that are created and written out, use this buffer.

pfx
= PZE if the unit is to be rewound before each header label is read or written and after each trailer label is read or written.
= PTW if the unit is to be rewound and unloaded after the trailer label is read or written.
= MZE if all repositioning is to be omitted.
= MON if both repositioning and verification or creation are to be omitted.

erret
is the location of an error routine, written by the user. If *erret* is left blank (zero) and an error is encountered, IOCS pauses and allows the operator to exercise his discretion as to the disposition of the invalid label.

mode
is an expression of the form A+4*B, where:
A = 1 for binary files, 2 for BCD files
B = 1 for input files, 2 for output files

ret1 and ret2
are the locations to which control is returned after the specified operation has been completed. The return taken varies with the conditions encountered.

The *mode* bits in this calling sequence override the corresponding bits in the label control block. Either or both of them may be left blank, however, and the label control block is then left unchanged.

Label System Operations

The following sections describe the four operations provided by IOLS. The procedures used for verifying and creating labels are described later, in the section "Label Verification and Creation." The label control block is modified according to the specifications of the calling sequence before any of the following operations is executed.

Open the File (ILOPN)

The IOLS operation code ILOPN may be used to open labeled or unlabeled, input or output files. The steps taken for each kind of file are detailed below.

LABELED INPUT FILE

1. The unit is repositioned according to the specifications of the calling sequence.

2. The header label is read into core storage and verified. If the label is invalid, the invalid-label return *(erret)* is taken.

3. The file mark following the header label is passed over.

4. If the label control block indicates that a checkpoint follows the header label, this checkpoint and the file mark following it are passed over.

5. Control is returned to *retl*.

UNLABELED INPUT FILE

1. The unit is repositioned according to the specifications of the calling sequence.

2. If the label control block indicates that the reel begins with a checkpoint, this checkpoint and the file mark following it are passed over.

3. Control is returned to *retl*.

LABELED OUTPUT FILE

1. The unit is repositioned according to the specifications of the calling sequence.

2. If the unit has been rewound, the existing header label is read into core storage to verify that the retention period has expired (this process is described in "Output Unit Header Label" in the section "Label Verification and Creation"). If the retention period has not expired, the file is presumably still valid and should not be overwritten. If the user attempts to open an output file with a reel containing a still-valid file, operator intervention is required.

3. The unit is backspaced over the old header label.

4. A new header label is created from the data in the label control block and is written out.

5. A file mark is written.

6. If the label control block specifies a checkpoint at the beginning of each reel, this checkpoint is written, followed by a file mark.

7. Control is returned to *retl*.

UNLABELED OUTPUT FILE

1. The unit is repositioned according to the specifications of the calling sequence.

2. If the unit has been rewound, the existing header label is read into core storage to verify that the retention period has expired. If the retention period has not expired, the file is presumably still valid and should not be overwritten. If the user attempts to open an output file containing a still-valid file, operator intervention is required.

3. The unit is backspaced over the header label.

4. If the label control block specifies a checkpoint at the beginning of the reel, this checkpoint is written, followed by a file mark.

5. Control is returned to *retl*.

If the file was already open, control is returned to *retl*.

Switch Reels (ILRSW)

The IOLS operation code ILRSW may be used to switch reels on an input file, either labeled or unlabeled, after a file mark has been read.

LABELED INPUT FILE

1. If the label is nonstandard and no nonstandard processing routine is specified, the label control block is examined to determine whether or not the file is multi-reel.

2. The trailer label is read into core storage and verified. If the label is invalid, the invalid-label return *(erret)* is taken.

3. If the label indicates that the end of the data file has been reached, no reel switching is necessary. The file mark following the trailer label is passed over and control is returned to *retl*. If the trailer indicates the end of the reel, the unit is repositioned according to the specifications of the calling sequence. If the unit is not to be rewound, the file mark following the trailer label is passed over.

4. The IOLS determines the alternate unit from the label control block. If no alternate is specified, a message is typed requesting that a new reel be mounted on the same unit.

5. The alternate unit or the new reel on the same unit is rewound.

6. The header label on the new reel is read into core storage and verified. If the label is invalid, the invalid-label return *(erret)* is taken.

7. The file mark following the header label is passed over.

8. If the label control block indicates that a checkpoint follows the header label, this checkpoint and the file mark following it are passed over.

9. Control is returned to *ret2*.

UNLABELED INPUT FILE

1. If the label control block specifies that the file is single-reel, control is returned to *ret1*.

2. If the label control block specifies that the file is multi-reel, the unit is repositioned according to the specifications of the calling sequence.

3. The IOLS determines the alternate unit from the label control block. If no alternate is specified, a message is typed requesting that a new reel be mounted on the same unit.

4. The alternate unit or the new reel on the same unit is rewound.

5. If the label control block indicates that the reel begins with a checkpoint, this checkpoint and the file mark following it are passed over.

6. Control is returned to *ret2*.

If the file is closed, the program error return *(erret)* is taken.

Force End-of-Reel Condition (ILFER)

The IOLS operation code ILFER may be used to switch reels on labeled or unlabeled, input or output files. The steps taken for each kind of file are detailed below:

LABELED INPUT FILE

1. The unit is repositioned according to the specifications in the calling sequence.

2. The IOLS determines the alternate unit for the file from the label control block. If no alternate unit is specified, a message is typed requesting that a new reel be mounted on the same unit.

3. The alternate unit or the new reel on the same unit is rewound.

4. The header label on the new reel is read into core storage and verified. If the label is invalid, the invalid-label return *(erret)* is taken.

5. The file mark following the header label is passed over.

6. If the label control block indicates that a checkpoint follows the header label, this checkpoint and the file mark following it are passed over.

7. Control is returned to *ret1*.

UNLABELED INPUT FILE

1. The unit is repositioned according to the specifications of the calling sequence.

2. The IOLS determines the alternate unit for the file from the label control block. If no alternate is specified, a message is typed requesting that a new reel be mounted on the same unit.

3. The alternate unit or the new reel on the same unit is rewound.

4. If the label control block indicates that the reel begins with a checkpoint, this checkpoint and the file mark following it are passed over.

5. Control is returned to *ret1*.

LABELED OUTPUT FILE

1. A file mark is written.

2. An end-of-reel trailer label is created, using data from the label control block. This label is written, followed by a file mark.

3. The unit is repositioned according to the specifications of the calling sequence.

4. The IOLS determines the alternate unit for the file from the label control block. If no alternate unit is specified, a message is typed requesting that a new reel be mounted on the same unit.

5. The alternate unit or the new reel on the same unit is rewound.

6. The existing header label is read into core storage to verify that the retention period has expired. If the retention period has not expired, the file is presumably still valid and should not be overwritten. If the user attempts to open an output file with a reel containing a still-valid file, operator intervention is required.

7. The unit is backspaced over the old header label.

8. A new header label is created from the data in the label control block and is written out.

9. A file mark is written.

10. If the label control block specifies a checkpoint at the beginning of each reel, this checkpoint is written, followed by a file mark.

11. Control is returned to *ret1*.

UNLABELED OUTPUT FILE

1. A file mark is written.

2. The unit is repositioned according to the specifications of the calling sequence.

3. The IOLS determines the alternate unit for the file from the label control block. If no alternate unit is specified, a message is typed requesting that a new reel be mounted on the same unit.

4. The alternate unit or the new reel on the same unit is rewound.

5. The existing header label is read into core storage to verify that the retention period has expired. If the retention period has not expired, the file is presumably still valid and should not be overwritten. If the user attempts to open an output file with a reel containing a still-valid file, operator intervention is required.

6. The unit is backspaced over the old header label.

7. If the label control block specifies a checkpoint at the beginning of each reel, this checkpoint is written, followed by a file mark.

8. Control is returned to *retl*.

If the file is closed, the program error return (*erret*) is taken.

Close the File (ILCLS)

The IOLS operation code ILCLS may be used to close labeled or unlabeled, input or output files. The steps taken for each kind of file are detailed below.

LABELED INPUT FILE

1. The unit is repositioned according to the specifications of the calling sequence.
2. Control is returned to *retl*.

UNLABELED INPUT FILE

1. The unit is repositioned according to the specifications of the calling sequence.
2. Control is returned to *retl*.

LABELED OUTPUT FILE

1. A file mark is written.
2. An end-of-file trailer label is created, using data from the label control block. This label is written, followed by a file mark.
3. The unit is repositioned according to the specifications of the calling sequence.
4. Control is returned to *retl*.

UNLABELED OUTPUT FILE

1. A file mark is written.
2. The unit is repositioned according to the specifications of the calling sequence.
3. Control is returned to *retl*.

If the file is closed, the unit is repositioned according to the specifications of the calling sequence and control is returned to *retl*.

Label Verification and Creation

To fill the label verification and creation requirements of the four major operations, IOLS treats five distinct kinds of labels. It verifies header and trailer labels from input files, checks the header labels on files that are about to be overwritten to ensure that the retention period has expired, and creates header and trailer labels for output files. Each of these procedures is described below.

Input File Header Label

Whenever an input file is opened and at the beginning of each reel of multi-reel file, the header label is verified to ensure that the correct reel is being processed. The following conditions must be met if the label of an input file is to be considered valid:

1. The label identifier must be 1HDRb.
2. The file creation date, in word LCRET of the label control block, must agree with the creation date in the label, Field 3. This condition is considered to have been met if the label control block word is zero.
3. The file identification, in words LCID1 and LCID2 of the label control block, must agree with the file identification in the label, Field 4. This condition is considered to have been met if the label control block words are both zero.
4. The file serial number, in words LCFSN of the label control block, must agree with the file serial number in the label, Field 5. This condition is considered to have been met if the label control block word is zero.
5. The reel sequence number, in word LCRSN of the label control block, must agree with the reel sequence number in the label, Field 7. This condition is considered to have been met if the label control block word is zero.

Unless all of the above conditions are satisfied, an invalid-label return (*erret*) is taken. The invalid-label return is described in the section "Error Procedures" in this chapter.

If any of the control information (i.e., check sum indicator, block sequence indicator, recording mode indicator, or checkpoint indicator) in a valid label contradicts the information found in the label control block, the label information replaces that in the label control block. If the unit is rewound, the block count in word LBCSN of the label control block is set to zero.

Input File Trailer Label

At the end of each file and at the end of each reel of a multi-reel file, the trailer label is inspected. The following conditions must be met if this label is to be considered valid:

1. The label identifier must be either 1EOFb or 1EORb.
2. The block count, in Field 20 of the label, must agree with the number of blocks read, in word LCBSN of the label control block. This condition is considered to have been met if the label control block word is zero.

If either of the above conditions is not met, an invalid-label return (*erret*) is taken.

Output Unit Header Label

Through the use of a System Monitor assembly parameter, each installation is allowed to specify whether or not the existing header labels on outputs units are to be checked for retention information before writing begins. This step has been included in the descriptions of opening and forcing the end-of-reel condition on output files. All programs should use IOLS for output reel handling, even if the files being created do not contain labels.

The following conditions must be met if a unit is to be used:

1. The label identifier must be 1HDRb.

2. The retention period must have expired. This is verified by checking the retention period (Field 2 of the label) and the file creation date (Field 3 of the label) against the current date, in location S.SDAT+1.

If either condition is not met, a message is typed requesting operator action. The operator may choose to use the reel, to replace the reel with another (on which the label is also verified), or to terminate the job.

Output File Header Label

When ILOPN or ILFER are used on labeled files, header labels are created and written. These labels are created in the following manner:

1. The label identifier is 1HDRb.

2. The following information is inserted in the label from information contained in the label control block:

Retention Period—Field 2—from word LCRET
File Identifier—Field 4—from words LCID1 and LCID2
File Serial Number—Field 5—from word LCFSN
Reel Sequence Number—Field 7—from word LCRSN
Check Sum Indicator—Field 10—from word LCCON
Block Sequence Indicator—Field 11—from word LCCON
Recording Mode Indicator—Field 12—from word LCCON
Checkpoint Indicator—Field 19—from word LCCON
Block Count—Field 20—from word LCBSN

3. The following information is also placed in the label:

Creation Date—Field 3—from location S.SDAT+1
Reel Serial Number—Field 6—from the previous header label on the same unit
Density Indicator—Field 9—always 0
Recording Technique Indicator—Field 13—always 6
Data Processing Technique Indicator—Field 14—always 6
Creating System—Field 15—always 7040

4. Fields 8, 16, 17, 18, and 21 through 27 are always blank. If the unit is rewound, the block count in word LCBSN of the label control block is set to zero.

Output File Trailer Label

When ILFER or ILCLS is used on labeled files, trailer labels are created and written on the old output unit. These labels are created in the following manner:

1. If the operation is ILFER, the label identifier is 1EORb. If the operation is ILCLS, it is 1EOFb.

2. The information that was placed in the header label is also placed in the trailer label.

3. The block count from word LCBSN is placed in the block count field of the label, Field 20.

Nonstandard and Additional Label Processing

The label control block can specify the location of a routine, written by the user, that provides label processing in addition to or instead of the standard IOCS routines. If nonstandard or additional processing is specified for the labels on a given file, the programmer must provide for the processing of all of the different kinds of labels that can appear on that file, except output unit header labels that are to be overwritten.

A nonstandard label may have either a rearrangement of the standard label format or an entirely different format. The IOLS still performs all of the actions associated with each IOLS operation, except label verification and creation. Control is transferred to the user's routine whenever verification or creation of a label is required.

Additional label processing involves the creation and verification of fields in the optional use portion of the standard 120-character label (positions 101-120). If additional label processing is specified, the user's routine is executed immediately after the IOLS creation or verification routine. Although provision must be made for processing each kind of label that can appear on the file, this provision may consist of no more than an immediate return without any actual processing. Such returns should always be made by a TRA 2,4.

Input File Header Labels: The following calling sequence is used to transfer control to the user's nonstandard or additional label processing routine for input file header labels:

```
TSX     lrt, 4
PZE     lcb, , label
```

where:

lrt
is the location of the entry point to the programmer's label processing routine.

lcb
is the location of the first word of the label control block.

label
is the location in core storage of the first word of the label.

If the programmer's routine finds the label valid, it should return control to IOLS by a TRA 2,4. If the label is invalid, the return is made by a TRA 3,4. Following nonstandard verification, the checkpoint indicator in the label control block must be set if a checkpoint is to be passed over.

Input File Trailer Labels: The following calling sequence is used to transfer control to the user's nonstandard or additional label processing routine for input file trailer labels:

```
TSX     lrt+1, 4
PZE     lcb, , label
```

If a nonstandard label processing routine finds that a label is an end-of-reel trailer (i.e., indicates that unit switching is required), it should return to IOLS by a

TRA 2,4. Since additional processing routines are only called for trailer labels with a 1EORb identifier, the TRA 2,4 return should be used, unless the additional processing routine finds other information which indicates that the end of the data file has been reached. At the end of the data file, both nonstandard and additional routines should return by a TRA 3,4.

Output File Header Labels: The following calling sequence is used to transfer control to the user's nonstandard or additional label processing routine for output file header labels:

```
        TSX         lrt+2, 4
        PZE         lcb, , label
```

Return should be made by a TRA 2,4.

Output File Trailer Labels: The following calling sequence is used to transfer control to the user's nonstandard or additional label processing routine for output file trailer labels:

```
        TSX         lrt+3, 4
        PZE         lcb, , label
```

Return should be made by a TRA 2,4.

The contents of index registers 1 and 2 must be saved and restored by the user's nonstandard or additional label processing routines.

Error Procedures

When any discrepancy is found in a label, either the calling program or the operator is allowed to make additional verification. If an error routine has been specified in the calling sequence to IOLS, this routine is called to process most label errors. If no routine has been specified, operator intervention is allowed for certain correctable errors. Certain errors can be handled only by the operator. See the IOLS decision table, Figure 15, for a summary of label error procedures. The following calling sequence is used by IOLS for error routines:

```
        TSX         erret, 4
        pfx         lcb, , label
        BRN         lcall  (used only if pfx = PZE
                           or MZE)
```

where:

pfx
= PZE if an error exists in the calling sequence to IOLS.
= MZE if the end of the medium was reached before IOLS could write an output file trailer label or while it was writing an output file header label.
= PON if a file mark or a redundancy occurred in the reading of a header label. If the error was a file mark, the label area is cleared before the error routine is called.
= MON if a file mark or a redundancy occurred in the reading of a trailer label. If the error was a file mark, the label area is cleared before the error routine is called.
= PTW if a header label was found to be invalid.
= MTW if a trailer label was found to be invalid.

lcb
is the location of the first word of the label control block for the file.

label
is the location of the first word of the label area.

lcall
is the two's complement of the location of the calling sequence to IOLS (given for pfx = PZE and MZE only).

The error routine is called for the following program errors:

1. A calling sequence in which *lcb* is zero.
2. A calling sequence in which *label* is zero.
3. A calling sequence in which *oper* is not a recognizable code.
4. A call for ILRSW on an unopened file or on an output file.
5. A call for ILFER on an unopened file or on a file on which no primary unit is specified.

Operation	Type of File	Trailer Label				Header Label			Always
		Redundant	Invalid	1EOF	1EOR	Redundant	Invalid	Valid	
ILOPN	Input					erret (PON)	erret (PTW)	ret1	
	Output					Operator Action Only		ret1	
ILRSW	Input	erret (MON)	erret (MTW)	ret1	Check Header	erret (PON)	erret (PTW)	ret2	
	Output								erret (PZE)
ILFER	Input					erret (PON)	erret (PTW)	ret1	
	Output				(written)	Operator Action Only			ret1
ILCLS	Input								ret1
	Output			(written)					ret1

Figure 15. The IOLS Decision Table

If no error routine has been specified and either an erroneous calling sequence (*pfx* = PZE) is encountered or the end of the medium is reached while writing an output label (*pfx* = MZE), the job is terminated. Any return to IOLS by a TRA 2,4 or a TRA 3,4 from a routine processing such errors also results in termination.

In the case of an input header label (*pfx* = PON or PTW), a return by a TRA 2,4 accepts the reel; while a return by a TRA 3,4 causes the reel to be unloaded. In the case of an input trailer label (*pfx* = MON or MTW), a return by a TRA 3,4 causes the end-of-data-file procedure to be executed, whereas a return by a TRA 2,4 causes the end-of-reel procedure to be executed.

If the error is in an input header or trailer label and no error return has been specified, an operator message is typed,' along with the label in error, the label expected, and instructions for operator options. The operator may, under sense switch control, exercise the same options that are available to the program. Discrepancies in output unit header labels (unexpired retention periods) always require operatior action.

If IOLS is called for a file on which no primary unit is specified in the label control block (i.e., if this field is zero), the call is ignored if the operation is ILOPN or ILCLS, and results in an end-of-data-file return if the operation is ILRSW.

1401 Input/Output Control Program (IOCP)

The 1401 Input/Output Control Program (IOCP) has been written to provide users of IBM 7040/7044 Data Processing Systems with easy access to input/output units connected to the 7040/7044 through an on-line 1401. The IOCP is designed to allow such input/output devices to be used as though they were connected to the 7040/7044 by a 1414 Input/Output Synchronizer.

The IOCP stays in a ready loop, awaiting a select instruction from the 7040/7044. When such an instruction is executed, it is brought into the 1401 and translated into the 1401 instructions necessary to perform the desired operation. The IOCP can read and/or write records on any of the following devices in accordance with instructions from the 7040/7044:

Up to six 729/7330 Magnetic Tape Units
A 1402 Card Read Punch
A 1403 Printer

Since the maximum size of magnetic tape records that can be written by the IOCP is a function of the core storage size of the 1401 and the amount of space occupied by the IOCP itself, the IOCP does not contain any error correction routines. Instead, it notifies the 7040/7044 of any errors that have occurred, and the program in control there must attempt error recovery by the appropriate procedures.

If a device connected to the 1401 is sensed when the IOCP is not in its ready loop, the first bit in the first byte of sense data is set on, indicating that the device is not ready.

The following sections describe the treatment of the various devices by the IOCP.

729/7330 Magnetic Tape

The following select instructions can be used for magnetic tape units connected to the on-line 1401:

Read BCD
Read Binary
Write BCD
Write Binary
Write Blank Tape
Write File Mark
Backspace One Record
Rewind
Rewind and Unload
Sense

The 7040/7044 and the 1401 are interlocked both while data is being transmitted to or from the 1401 and while it is being read from or written on the tape. If an error occurs, the IOCP does not attempt to correct it; instead, it turns on the channel A redundancy indicator.

The Write Blank Tape, Write File Mark, Backspace One Record, Rewind, and Rewind and Unload instructions interlock the 7040/7044 and the 1401 only long enough to allow the IOCP to interpret them. The 7040/7044 is then allowed to proceed with its processing, and the IOCP executes the appropriate 1401 instruction before returning to its ready loop. The Write Blank Tape instruction must be followed by a Reset and Load Channel instruction in the 7040/7044, although no data is transmitted. The IOCP issues an Erase Tape instruction, but the blank tape section is not actually written until the next Write instruction for the same tape unit is executed.

Tape records may not exceed 370 7040/7044 words in length. An attempt to write a record longer than this results in the loss of the additional words.

Unless the IOCP is out of the ready loop, there can be no bits in the first byte of sense data for a magnetic tape unit on the 1401. The tape is considered ready if the IOCP is ready.

1402 Card Reader

The following select instructions are allowed for the card reader of a 1402 Card Read Punch connected to the on-line 1401:

Read BCD
Read Binary
Sense

Except for the reading of the first card, the IOCP is always one card ahead of the 7040/7044. This allows the IOCP to transmit a card image as soon as one is requested, without having to wait for the actual read operation to take place. This results in a minimum of interlock time, since the only nonoverlapped operation is the transmission of the data from the 1401 to the 7040/7044. The 7040/7044 can then continue processing, while the IOCP reads another card before returning to its ready loop. Cards read without error are stacked in Pocket 1.

The 7040/7044 can test the status of the card currently held in the 1401 by executing a sense instruction on the card reader. If no card has yet been read, the sense instruction yields a unit-busy indicator and the 1401 proceeds to read a card.

The significance of the bits in the first byte of sense data is:

S bit	0
1 bit	0
2 bit	0
3 bit	Unit is busy
4 bit	No more cards to be read (end of file)
5 bit	Data is binary
No bits	Data is BCD

There can be no combinations of these bits.

1402 Card Punch

The following select instructions are allowed for the card punch of a 1402 Card Read Punch connected to the on-line 1401:

Write BCD
Write Binary
Sense

The 7040/7044 and the 1401 are interlocked only during the transmission of data to the 1401. The 7040/7044 can then continue processing, and the IOCP punches the card before returning to its ready loop. Cards punched without error are stacked in pocket 4.

There can be no bits in the first byte of sense data for the punch. If the IOCP is ready, the punch is assumed to be ready.

1403 Printer

The following select instructions are allowed for the 1403 Printer connected to the on-line 1401:

Write BCD
Control
Sense

The 7040/7044 and the 1401 are interlocked only during the transmission of data to the 1401. The 7040/7044 can then continue processing, and the IOCP prints the line before returning to its ready loop.

A control select of the printer causes one word to be transmitted to the 1401. The IOCP uses the first six bits of this word as a carriage control character for a Control Carriage instruction. The following check is made of this character:

J is ignored
K is changed to J
L is changed to K

The only bit that can be on in the first byte of sense data for the 1403 Printer is bit 3, which indicates that the printer is busy.

Invalid Select Instructions

If the IOCP receives a select instruction that it cannot interpret, the 7040/7044 and the 1401 interlock, while the IOCP tries to reread the select instruction. If the reread is successful, processing continues normally. If the rereading fails to produce an instruction that the IOCP can accept, it enters a loop and waits for the operator to set a sense switch on the 1401 console.

If sense switch G is turned on, the IOCP halts. The operator can then press the START-RESET button, causing the invalid select instruction to be discarded. Processing resumes in both computers.

If sense switch F is turned on, the IOCP itself discards the invalid select instruction. Processing then resumes in both computers.

Index

A page number in *italics* indicates that the designated reference is of particular importance.