File Number S360-30
Form C24-5037-1

DOS

IBM

Systems Reference Library

IBM System/360
Disk Operating System
Supervisor and Input/Output Macros

This reference publication contains planning information about the Input/Output Control System Macro Instructions and the Supervisor Macro Instructions for use with the IBM System/360 Disk Operating System (DOS). IBM publications that provide related information are:

IBM System/360 Principles of Operation, Form A22-6821;

IBM System/360 Disk Operating System, Data Management Concepts, Form C24-3427;

IBM System/360 Disk Operating System, System Control and System Service Programs, Form C24-5036;

IBM System/360 Disk and Tape Operating Systems, Assembler Specifications, Form C24-3414;

IBM System/360 Disk Operating System, Basic Telecommunications Access Method, Form C30-5001.

For titles and abstracts of other associated publications, see the IBM System/360 Bibliography, Form A22-6822.

PREFACE

This reference publication is intended as a guide for the programmer planning to use the DOS Input/Output Control System macro instructions and Supervisor macro instructions. The publication is divided into seven sections. The first two sections introduce concepts and terminology. The third section describes the LIOCS imperative macro instructions, while the fourth section discusses the two types of LIOCS declarative macro instructions. The fifth section contains information about processing with PIOCS. The sixth section describes the Supervisor-Communication Macros and the procedures for checkpointing a program. The seventh section discusses additional macro instructions used in program linkage.

The material in this publication is presented with the assumption that the reader has experience with computer systems and basic programming concepts and techniques (or has completed basic courses of instruction in these areas).

The user should be familiar with the publications listed on the front cover of this manual.

Second Edition, November 1966

This edition, Form C24-5037-1, is a major revision of, and obsoletes, Form C24-5037-0 and Technical Newsletters N24-5111 and N24-5148. Changes are indicated by a vertical line to the left of the affected text and to the left of affected parts of figures. A dot (•) next to a figure title or page number indicates that the entire figure or page should be reviewed.

Significant changes and additions to the specifications contained in this publication will be reported in subsequent revisions or Technical Newsletters.

Requests for copies of IBM publications should be made to your IBM representative or to the IBM branch office serving your locality.

A form is provided at the back of this publication for readers' comments. If the form has been removed, comments may be addressed to IBM Corporation, Programming Publications, Endicott, New York 13760

SYSTEM CONFIGURATION

This section presents the minimum system configuration required to operate the Disk Operating System and features, in addition to the minimum that can be supported. The system control programs must always be present to execute other programs.

MACHINE REQUIREMENTS

Minimum features required:

16K bytes of main storage (32K bytes are required for multiprogramming and/or telecommunications, COBOL, assembler with tape or disk work file variants, and for assigning system input/output files to disk).
Standard instruction set. See Note 1.
One I/O channel (either multiplexor or selector). See Note 2.
One Card Reader (1442, 2501, 2520, or 2540). See Note 3.
One Card Punch (1442, 2520, or 2540). See Note 3.
One Printer (1403, 1404, or 1443). See Note 3.
One 1052 Printer-Keyboard.
One 2311 Disk Storage Drive.

Note 1: Language translators may require extended instruction sets.

Note 2: Telecommunications requires a multiplexor channel and at least one selector channel.

Note 3: One 2400-series magnetic tape unit may be substituted for this device. (7- or 9-track. If 7-track tape units are used, the data-convert feature is required, except when substituted for a printer.)

Additional features supported:

Timer Feature.
Simultaneous Read-while-Write Tape Control (2404 or 2804).
Any channel configuration up to one multiplexor channel and six selector channels.
Tape Switching Unit (2816).
Storage Protection Feature (required for multiprogramming).
Universal Character Set.
Additional main storage up to 16,777,216 bytes.

Problem programs can request I/O operations on the following devices:

1. 1442 Card Read Punch
2. 2501 Card Reader
3. 2520 Card Read Punch
4. 2540 Card Read Punch
5. 1403 Printer
6. 1404 Printer (for continuous forms only)
7. 1443 Printer
8. 1445 Printer
9. 1052 Printer-Keyboard. It is used for operator communication.
10. 2671 Paper Tape Reader
11. 2311 Disk Storage Drive
12. 2321 Data Cell Drive
13. 2401, 2402, 2403, 2404, and 2415 Magnetic Tape Units.
14. 1285 Optical Reader* (maximum of 8 are supported)
15. 1287 Optical Reader* (maximum of 8)
16. 1030 Data Collection System
17. 1050 Data Communication System
18. 1060 Data Communication System
19. 2260 Low Cost Display Station
20. AT&T 83B3 Selective Calling Stations
21. AT&T Teletypewriter Terminal, Models 33 and 35
22. Western Union Plan 115A Outstations
23. 2740 Communication Terminal
24. 7770 and 7772 Audio Response Units

* Programming specifications for using these devices may be used for planning purposes only. Source programs must not contain instructions for these devices until the Disk Operating System includes the appropriate programming. An MNOTE

stating IMPROPER DEVICE will appear if coding for these devices is included in a source program.

The preceding devices (16 through 23) are attached by means of a private, leased, or common-carrier network to the multiplexor channel through a 2701 Data Adapter Unit, 2702 or 2703 Transmission Control Unit. When the 2701, 2702, or 2703 is attached to the multiplexor channel, burst-mode devices (magnetic tape and DASD) must be attached to a selector channel.

MACRO SIMILARITIES

Macro similarities between Basic Programming Support (BPS), Basic Operating System (BOS), Tape Operating System (TOS), and Disk Operating System (DOS) are as follows:

- Imperative macro instructions and Supervisor Communication macro instructions available for TOS have identical expansions for DOS.
- File definition macro instructions available for the TOS have identical expansions for DOS.

Symbolic programs written for BPS and BOS can be assembled into the functional equivalents for DOS. However, the DTFSR macro instruction substantially prolongs program preparation time when used on DOS.

Certain register parameters for BPS and BOS are parenthesized in DOS. In general, DOS will accept the parameters without the parentheses. For example, in DOS the correct format is IOREG=(r). DOS will accept the BOS parameter IOREG=r.

No change in register usage conventions is planned for BOS or BPS. However, to avoid compatibility problems resulting from transition between 8K and 16K support levels, installations using 8K support should observe the 16K register conventions indicated under Register Usage.

REGISTER USAGE

General registers 0, 1, 13, 14, and 15 have special uses, and are available to the programmer only under certain restrictions. Registers 2 through 12, however, are available without restriction and, to avoid the possibility of errors, these should be the registers used by the programmer.

Registers 0, 1

Logical IOCS macros, the Supervisor macros, and other IBM-supplied macros use these registers to pass parameters. Therefore, these registers may be used without restriction only for immediate computations, where the contents of the register are no longer needed after the computation. If the programmer uses them, he must either save their contents himself (and reload them later) or finish with them before IOCS uses them.

Register 13

Control program subroutines, including logical IOCS, may use this register as a pointer to the 72-byte, double-word aligned save area. Most users will find it convenient to set the address of the save area in register 13 at the beginning of each program phase, and to leave it unchanged thereafter.

Registers 14, 15

Logical IOCS uses these two registers for linkage. Register 14 contains the return address (to the problem program) from DTF routines, called programs, and user's subroutines. Register 15 contains the entry point into these routines, and is used as a base register by the OPEN, CLOSE, and certain DTF macros. IOCS does not save the contents of these registers prior to using them. If the programmer uses them, he must either save their contents himself (and reload them later) or finish with them before IOCS uses them.

COMPATIBILITY OF THE ORIGINAL AND THE PRESENT DOS

- User programs written for the original DOS can be run in the present DOS batch-job environment with no changes and without recompilation.
- A LABADDR routine that builds user-standard labels in the IOCS area of main storage cannot be executed in a storage-protected environment and, hence, cannot be executed in a multiprogramming environment.
- Programs written for the original DOS can be run as background jobs in a multiprogramming environment with the following exception. The background program must have control of the interval timer feature when a program using the timer (SETIME macro instruction) is executed in a multiprogramming environment.
- Programs using PIOCS to process DASD files must OPEN the file if the program is being executed in a system that has DASD file protect. The channel program

for the DASD must begin with a Seek (X'07') command.

- Data files created for the original DOS can be used without exception.
- Programs checkpointed in the original DOS cannot be restarted in the present DOS.

PHYSICAL IOCS VS LOGICAL IOCS

The input/output control is considered to consist of two parts: physical IOCS (PIOCS) and logical IOCS (LIOCS). Physical IOCS controls the actual transfer of records between the external medium and main storage. It performs the functions of initiating the execution of channel commands and handling associated I/O interrupts. Physical IOCS consists of the following routines:

- Start I/O routine
- Interrupt routine
- Channel scheduler
- Device error routines.

These physical IOCS routines are part of the supervisor, which is permanently located in lower main storage while problem programs are being executed. The device error routine for SYSRES resides in the supervisor area. Other error routines are called into the transient area.

Logical IOCS performs those functions that a user needs to locate and access a logical record for processing. A logical record is one unit of information in a file of like units--for example, one employee's record in a master payroll file, one part-number record in an inventory file, or one customer account record in an account file. One or many logical records may be included within one physical record, for example, a physical tape record (gap to gap). The term logical IOCS refers to the routines that perform the following functions:

- Blocking and deblocking records
- Switching between I/O areas when two areas are specified for a file
- Handling end-of-file and end-of-volume conditions
- Checking and writing labels.

Logical IOCS uses physical IOCS to execute I/O commands whenever it determines that a transfer of data is required. For example, if a file consists of blocked records and a block has been read into main storage, logical IOCS makes each record in succession available to the user until the end of the block is reached. No physical IOCS is required. When logical IOCS determines that the last record in the block has been processed, it requests physical IOCS to start an I/O operation to transfer the next physical record (gap to gap) into main storage.

In Figure 1, only logical IOCS is required to make records 2 and 3 (and 5 and 6) available. Records 1-3 are already in main storage. Physical IOCS is also required to make record 4 available (records 4 through 6 are transferred in one block).

Logical IOCS macros (such as GET, PUT, READ, WRITE, etc) and physical IOCS macros (such as EXCP and WAIT) are available to the programmer for handling records. The logical IOCS macro routines cause all the functions of both logical and physical IOCS to be performed for the programmer. For example, when a GET instruction is issued, a logical record is made available for processing. Registers 0, 1, 14, and 15 are used by logical IOCS routines.

The physical IOCS routines are completely distinct from the routines used by logical IOCS to perform functions such as blocking and deblocking. They permit the problem program to use physical IOCS functions directly. To transfer a physical record (such as a DASD or tape record), the problem program issues an EXCP macro instruction (Execute Channel Program). This causes a request for data transfer to be handled by the channel scheduler, and program execution immediately continues with the next problem program instruction. However, the DASD or tape record will not be available in main storage until some later time. When the record is needed for processing, the program must test (WAIT macro instruction) to determine if the transfer has been completed. Physical IOCS uses registers 0 and 1.

The functions of physical and logical IOCS routines are shown in Figure 2.

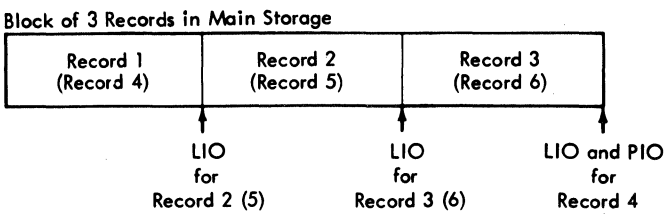


Figure 1. Physical IOCS vs Logical IOCS

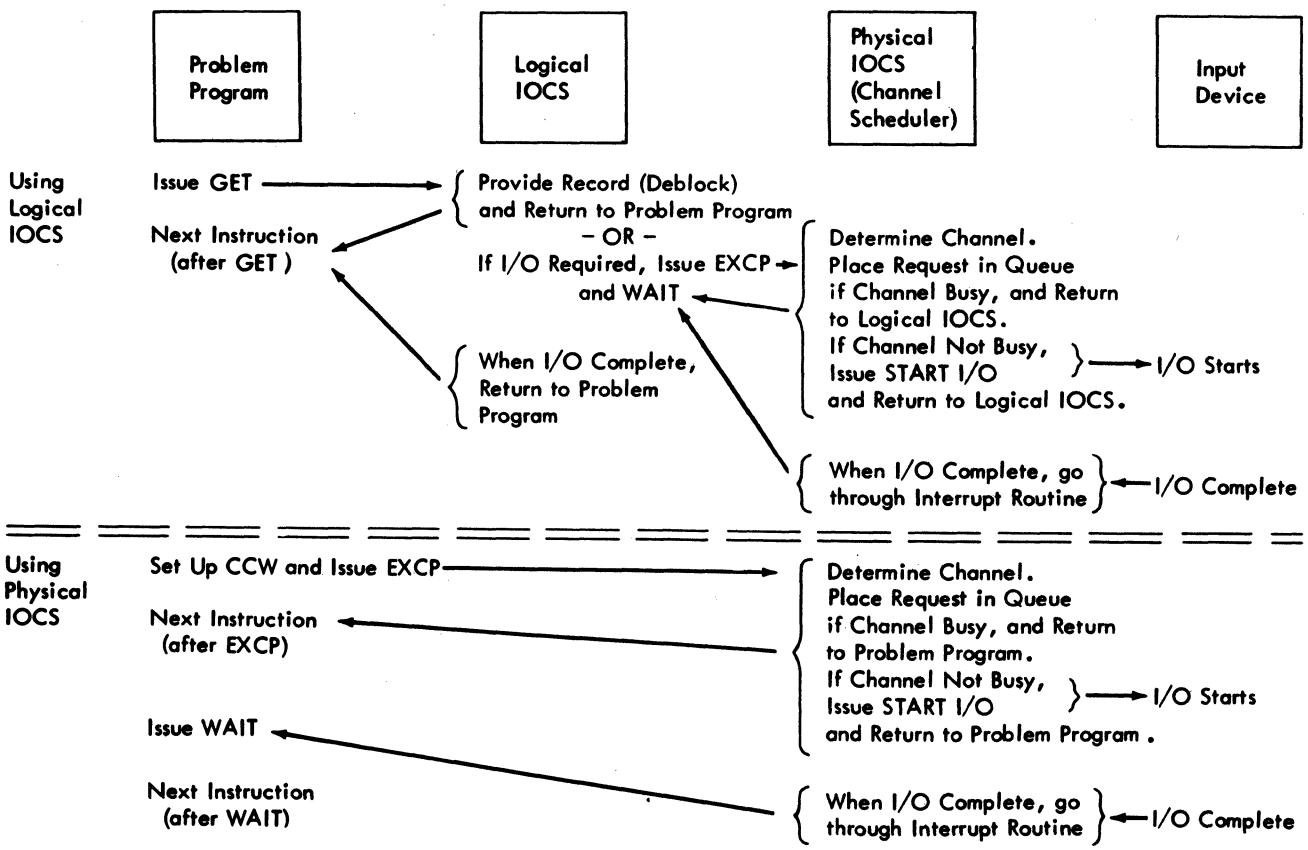


Figure 2. Retrieving a Record Using Logical IOCS (One I/O Area) or Physical IOCS

DASD File Protection

For a 32K or larger machine, logical and physical IOCS can provide DASD data file protection if the user requests it when he generates his system. The protection is on a cylinder basis for the 2311 and on a strip basis for the 2321. Thus, data files having the same cylinder/strip cannot be protected from one another. For complete protection, files should begin and end on cylinder/strip boundaries. File protection is ensured only if the labels involved are unexpired.

TYPES OF LIOCS PROCESSING

The logical IOCS routines process records in sequential order, in random order by the Direct Access Method (DAM), or randomly and sequentially by the Indexed Sequential File Management System (ISFMS). Sequential processing applies to all files in serial I/O devices (such as card reader, tape, printer, etc), and to records on the IBM 2311 disk or 2321 data cell when they are processed serially. The types of processing performed by DAM and ISFMS apply only to files of Direct Access Storage Device (DASD) records.

SEQUENTIAL PROCESSING

Sequential processing is used to read/write and process successive records in a logical file. For example, card records are processed in the order the cards are fed. Tape records are processed starting with the first record after a header label and continuing through the records to the trailer label. DASD records are processed starting with a beginning DASD address and continuing in order through the records on successive tracks and cylinders to the ending address.

A sequential file on DASD is contained within one or more sets of limits, which are specified by the Job Control XTENT cards. If the logical file consists of more than one set of limits, IOCS will automatically process each set as required by the user. The records within each set must be adjacent and contained within one volume (disk pack or data cell). The sets are not required to be adjacent or on the same volume. Sequential processing of a file written on DASD by the direct access method can be performed.

The basic macros used for sequential processing are GET and PUT. These instructions overlap data transfer and processing.

The extent of overlap depends on the user's I/O area assignment. In any case, when a GET or PUT has been executed, the transfer of data is complete before the instruction following the GET or PUT is executed.

DIRECT ACCESS METHOD (DAM)

The Direct Access Method (DAM) provides a method of processing records contained on IBM 2311 or IBM 2321 that are usually organized in a random manner. It is important to note that DAM is a method of processing records and not an organizational method.

IOCS locates a DASD record for processing by referring to a record-location reference supplied by the problem program. The location reference consists of two parts: a track reference and a record reference. The track reference specifies the track (or the first of multiple tracks) to be searched for the record. The record reference may be the record key, if records contain key areas, or the record identifier (ID), which is in the count area of each DASD record. IOCS seeks the specified track and searches for the record on that track, or on the succeeding tracks in the cylinder.

The basic macros used for the direct access method of processing are READ and WRITE. Variations within these macros permit records to be read, written, updated, replaced, or added to a file. Thus, this method provides a means of maintaining a logical file in a random (or sequential) order. When a READ or WRITE instruction is executed, the actual I/O operation is either started or placed in a queue for later execution. When the record is required for processing, the program must test (WAITF macro) to ensure that the transfer is complete.

INDEXED SEQUENTIAL FILE MANAGEMENT SYSTEM

DASD records contained within an indexed sequential file may be processed in a random order or in sequential order by control information. Both orders use the control information of the records (such as employee number, part number, etc), which is available in the key area of each DASD record. Any record stored at any location in the logical file can be processed using the random method. The user supplies ISFMS with the key (control information) of the desired record. ISFMS searches for the record and makes it available for processing.

In sequential processing, ISFMS makes a series of records available. The records are available, one after the other in order by the control information (key) in the records. The first record to be processed is specified by the user. ISFMS retrieves the succeeding records (on demand) from the logical file, in key order, until the problem program terminates the operation.

ISFMS provides the facility to create an organized file and then add to, read from, and update records in that file. The file is organized from records that have been pre-sorted by their control information. As the records are loaded onto DASD, ISFMS constructs indices for the logical file. The indices will permit individual records to be found in subsequent processing operations. The indices are created in such a way that records can be retrieved randomly or sequentially. If records are added to the file at a later date, ISFMS updates the indices to reflect the new records.

The basic macros used for processing the indexed sequential files are READ/WRITE and GET/PUT. READ and WRITE are used for random operations, and GET and PUT are used for sequential operations. A READ or WRITE instruction in the problem program causes the I/O operation to be started or placed in a queue, and execution of the problem program continues. When an instruction later in the program requires that the transfer of data be complete, a test must be made. A WAITF macro is provided for the test. When a GET or PUT instruction for a record is executed, the transfer of data is completed before the next instruction in the problem program is executed.

BASIC TELECOMMUNICATIONS ACCESS METHOD

Disk Operating System provides the facility to communicate with remote terminals by using the Basic Telecommunications Access Method (BTAM). The BTAM publication listed on the front cover of this manual provides a general description of the available Telecommunications facilities and specific information on the imperative macro instructions, DTF's, and modules used with BTAM.

A definition of macro is 'of or involving large quantities'. For one macro instruction, many instructions may be assembled. Thus, the system derives its name.

The macro system is composed of two basic parts:

- Macro Definitions--General routines written as source statements and stored in the Assembler Sub-library of the Source-Statement Library.
- Source-program macro instructions:
  1. Imperative Input/Output Control Macro Instructions tell what I/O operation is desired. For example, in Appendix C GET indicates that the user wants to obtain a card record.
  2. Supervisor Communication Macro Instructions communicate with the Supervisor and provide access to the communication region.

For processing with IOCS, two additional macro instructions are used:

3. Declarative Logic Module Generation Macro Instructions (used with LIOCS) provide information about the type of module to be generated. A module is an object code routine which can handle the conditions specified in the module generation macro. For example, in Appendix C the CDMOD generates a module to handle card input on a 2540 using a work area.
4. Declarative DTFxx Macro Instructions (used with the LIOCS and PIOCS) define the characteristics of the specific file to be processed. The information in the macro instruction is assembled into a DTF table. For example, in Appendix C the DTFCD macro instruction used specifies that the symbolic unit containing the file is SYS004, that the file uses a work area and an I/O area called A1, and that control should be given to EOFCD when the last card is read.

A direct relationship exists between these parts. During assembly, the macro instruction specifies which macro definition is to be called. The macro definition is extracted, tailored, and inserted into the program as shown in Figure 3. The

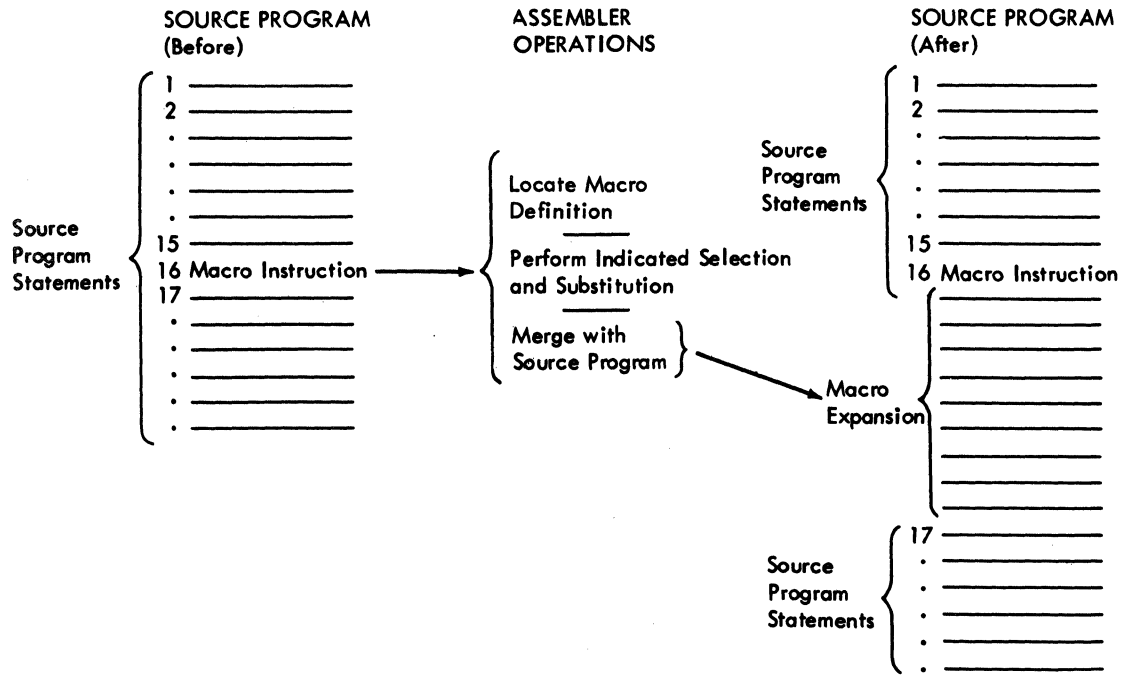


Figure 3. Schematic of Macro Processing

tailoring is accomplished by a selection and substitution process using the general information in the macro definition and the specific information in the macro instruction. The insertion is a module, a table, or a small in-line routine and is called the macro expansion.

After the insertion is made, the complete program consists of both source program statements and assembler language statements generated from the macro definition. In subsequent phases of the assembly, the entire program is processed to produce the machine-language program.

IBM provides a number of prewritten macro definitions and specifies the macro instructions that can be used by the programmer to use the definitions. Other macro definitions can be written by the user. See the Assembler publication listed in the abstract of this manual for information on this.

The IBM-supplied macro instructions that are explained in this publication are organized in four categories:

- Imperative LIOCS I/O Control Macro Instructions
- Declarative LIOCS DTF and I/O Module Generation Macros
- PIOCS Macro Instructions
- Supervisor Communication Macro Instructions

Self-Relocation and IOCS

To make LIOCS and PIOCS Imperative and Supervisor Communication macro instructions self-relocating the user must:

1. use the OPENR and CLOSER macro instructions, and he must
2. use register notation within his macro instructions.

See the discussion on Register Notation in this publication.

DTF MACRO

Whenever logical IOCS imperative macro instructions (GET, PUT, READ, WRITE, etc) are used in a program to control the input/output of records in a file, that file must be defined by a declarative macro instruction, called a DTF. The DTF macro instruction describes the characteristics of the logical file, indicates the type of processing to be used for the file, and specifies the main-storage areas and routines used for the file.

For example, if a GET is issued, the file definition supplies such factors as:

- Record type and length
- Input device from which the record is to be retrieved
- Address of the main-storage area where the record is to be located for processing by the problem program.

Ten different file-definition declarative macro instructions are available for defining files processed by LIOCS. One is available for magnetic tape or DASD files processed by PIOCS.

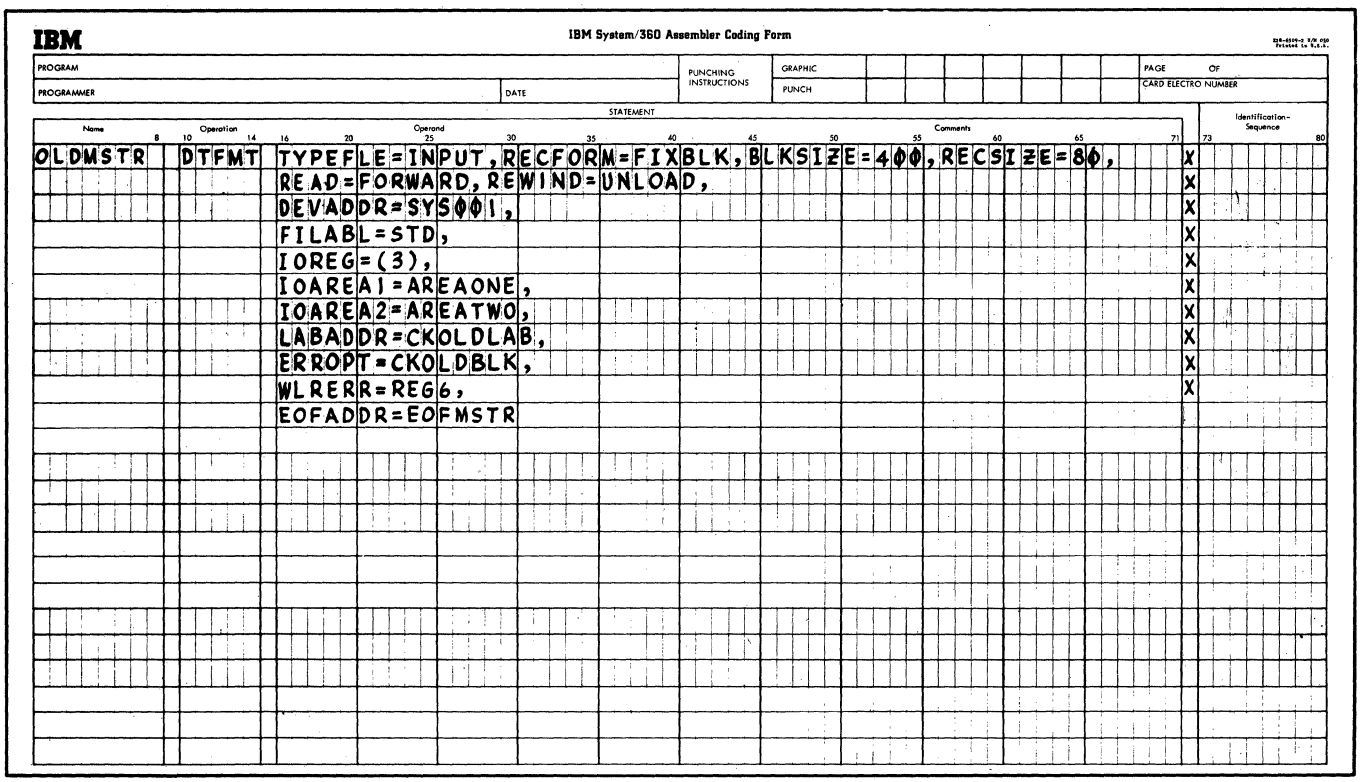
Figure 4 contains an example of a DTF. For LIOCS operations, the file-definition macro instructions used depend on the type of processing that will be performed for the file.

SEQUENTIAL PROCESSING: This applies to input/output files in serial devices or to 2311 or 2321 DASD when records are processed sequentially. The following macros are used for sequential processing:

Macro Instruction	Define the file for a:
DTFSR	Serial type device
DTFCD	Card Device
DTFMT	Magnetic tape
DTFPR	Printer
DTFCN	Console
DTFSD	Sequential DASD
DTFPT	Paper Tape Reader
DTFOR	Optical Reader

DIRECT ACCESS METHOD: Whenever a logical DASD file is to be processed randomly, the DTFDA is used.

INDEXED SEQUENTIAL SYSTEM: Whenever a logical DASD file is to be organized or processed by the indexed sequential file management system (ISFMS), the DTFIS is used.



IBM System/360 Assembler Coding Form

Name	Operation	Operand	Comments	Identification-Sequence
OLDMSTR	DTFMT	TYPEFLE=INPUT,RECFORM=FIXBLK,BLKSIZE=400,RECSIZE=80,		X
		READ=FORWARD,REWIND=UNLOAD,		X
		DEVADDR=SYS001,		X
		FILABL=STD,		X
		IOREG=(3),		X
		IOAREA1=AREAONE,		X
		IOAREA2=AREATWO,		X
		LABADDR=CKOLDLAB,		X
		ERROPT=CKOLDBLK,		X
		WLRERR=REG6,		X
		EOFADDR=EOFMSTR		

Figure 4. Sample DTFMT Macro Instruction

PHYSICAL IOCS PROCESSING: When PIOCS macro instructions (EXCP, WAIT, etc) are used for a file, the DTFPH macro instruction is required if standard labels are to be checked or written on a DASD or magnetic tape file, or if the DASD file is file protected.

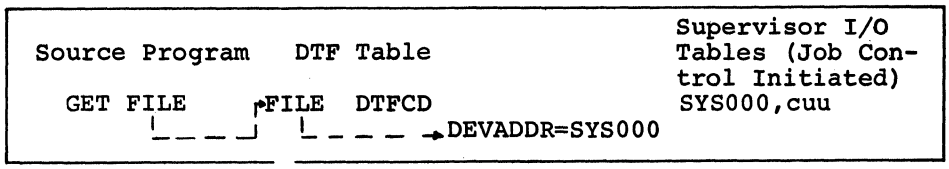
A DTFxx macro instruction generates a DTF table that contains indicators and constants describing the file. The user can reference this table by using the symbol Filename+constant or Filenamex, where x is a letter. When such a reference is necessary, the constant or letter is specified in the text. When referencing the DTF table, the user must ensure addressability through the use of an A-type constant, or through reference to a base register.

SYMBOLIC UNIT ADDRESSES

In each of the DTF macro instructions, except those for DTFDA, DTFIS, and DTFPH MOUNTED=ALL files, the user must specify a symbolic unit name in the DEVADDR=SYSnnn entry. This symbolic unit name is also used in the Job Control ASSGN statement to assign an actual I/O device address to the file. For DASD files the symbolic unit name is supplied in the Job Control XTENT statement.

The symbolic unit name of a device is chosen by the programmer from a fixed set of symbolic names. He writes his program considering only the device type (tape, card, etc) of his file. At execution time, the actual physical device is determined and assigned to a given symbolic unit. For instance, a programmer can write a program which processes tape records and can call the tape SYS000. At execution time the operator (using ASSGN) assigns any available tape drive to SYS000.

The relationship between the source program, the DTF table, and the Job Control I/O assignment is shown in the following chart.



The fixed set of symbolic unit names differs for batch or background jobs and for foreground programs. No other names can be used. For batch or background jobs the names are:

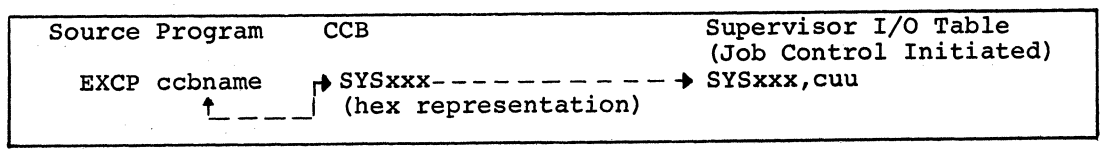
SYSRDR	Card reader, magnetic tape unit, or disk extent used for Job Control statements.
SYSIPT	Card reader, magnetic tape unit, or disk extent used as the input unit for programs.
SYSPCH	Card punch, magnetic tape unit, or disk extent used as the main unit for punched output.
SYSLST	Printer, magnetic tape unit, or disk extent used as the main unit for printed output.
SYSLOG	Printer-keyboard used for operator messages and logging Job Control statements. Can also be assigned to a printer.
SYSRES	System residence area on a disk drive (for DTFPH only).
SYS000-SYS244	All other background units in the system.

The permissible names for foreground programs are given in the list that follows. (Note that although some of the same names can be used in both background and foreground programs, separate device assignments are required for the programs.)

SYSLOG	Printer-keyboard used for operator messages and logging Job Control statements. Can also be assigned to a printer. The same device is used by background and foreground programs and the control program.
SYS000-SYSnnn	All other foreground units in the system. A total of 245 numeric units can be specified in any given system (at system generation time). Sequential names beginning with SYS000 are used for each area in the multiprogramming environment.

Because DTFSR is used in DOS purely for compatibility with BOS, different symbolic unit names are available. The names are listed in the DTFSR discussion.

In physical IOCS, the symbolic unit name is specified in the CCB (as well as the DTFPH when used). The relationship between the source program and the Job Control I/O assignment is shown in the following chart.



LOGIC MODULE GENERATION MACRO INSTRUCTIONS

Each DTF except DTFCN and DTFSR must link to an IOCS logic module. These modules provide the necessary instruction to perform the input/output functions required by the problem program. For example, the module may read or write data, test for unusual input/output conditions, block or deblock records if necessary, or place logical records in a work area.

Some of the module functions are provided on a selective basis, according to the parameters specified in the xxMOD macro-instruction. The programmer has the option of selecting (or omitting) some of these functions according to the requirements of his program. The omission of some of these functions will result in smaller main storage requirements for a particular module.

Note: If the user issues an imperative macro, such as WRITE or PUT, to a module that does not contain that function, the job is terminated and a message is displayed.

A logical IOCS module is defined as a subset to another logic module if all the functions available in the subset module, plus additional functions, are also available in the latter module, defined as a superset module. For example:

Superset Module Functions	Subset Module Functions
Optional use of CNTRL macro	CNTRL macro cannot be used
Workarea and I/O Area Processing	I/O Area Processing Only
Support of Printer Overflow	No Printer Overflow Support
Read backward and forward on magnetic tape	Read forward Only

The relation between subsets and supersets is shown in diagrams at the end of the discussion of each module.

Some CDMOD, PRMOD, PTMOD, ISMOD, SDMOD, DAMOD, and MTMOD macro instructions correspond, respectively, to two or more DTFCD, DTFPR, DTFPT, DTFIS, DTFSD, DTFDA, and DTFMT macro instructions. The functions required by these DTF's are thus available in a single xxMOD macro instruction, even if the DTF's have slightly different parameters.

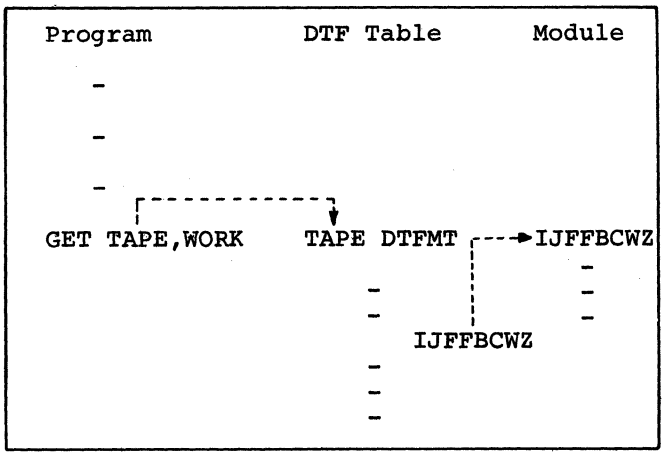
INTERRELATIONSHIPS OF THE MACRO INSTRUCTIONS

HOW THE IOCS MODULE IS LINKED WITH THE DTF TABLE

Regardless of the method of assembling logic modules and DTF's (i.e., with the main program or separately), a symbolic linkage will result between the DTF table and the logic module. The Linkage Editor will resolve these linkages at edit time.

The IOCS-module-DTF-table linkage is accomplished by generating a V-type address constant in the DTF table and a named CSECT in the logic module. To resolve this linkage, the linkage symbols must be identical. These linkage symbols are referred to as module names.

The following example illustrates the relationship of the program, the DTF, and the logic module. Imperative macros initiate the action to be performed on the file by branching to the logic module entry point generated in the DTF table.



TAPE is the name of the file. IJFFBCWZ is the name of the logic module.

GENERATION OF MODULE NAMES IN DTF TABLES AND LOGIC MODULES

A module name is generated in the DTF table by one of two methods. The user may explicitly specify the module name by entering it in the DTF parameter MODNAME=Name, or he may allow the macro definition that processes the DTF macro instruction to generate this name, as determined by the functions required by the DTF macro instruction.

Similarly, a module name is generated for a logic module by one of two methods. The user may explicitly specify the module name by supplying this name in the name field of the xxMOD macro instruction, or he may allow the macro definition that processes the xxMOD macro instruction to generate this name, as determined by the functions that the logic module will supply. The generated names will be referred to as standard module names. Information on standard module names follows discussions of the logic module generation macros.

SUBSET AND SUPERSET MODULE NAMES

When a DTF table is assembled (with the main program or separately), a module name is generated that reflects exactly the functions required by the DTF macro instruction.

If similar DTF's are assembled together, the functions required by the similar files will be collected by the macro definition during the assembly process, and one superset module name will be generated. For example, if in the same assembly one DTFCD table requires the CONTROL function and another DTFCD table does not, a CDMOD handling the CNTRL macro is able to process both DTFCD tables.

For example, an MTMOD with the CKPTREC option and the WORKA option can process a number of similar DTFMT tables requiring different functions. The following tables could be processed by a single MTMOD.

- No CKPTREC or WORKA
- CKPTREC and no WORKA
- WORKA and no CKPTREC
- Both CKPTREC and WORKA

Each logic module section contains a name list and indicates the structure of subset and superset names for the module.

EDITING LOGICAL IOCS PROGRAMS

The programmer has the option of either assembling DTF's and logic modules with his main program or assembling them separately for later linkage editing with the main program. In order to take full advantage of the linking facilities for DTF tables and logic modules, which are discussed in the following sections, the parameter SEPASMB=YES should be specified when DTF tables or logic modules are separately assembled.

Logical IOCS programs will always generate symbolic linkages between DTF tables and logic modules that will have to be resolved by the Linkage Editor at edit time. Furthermore, if DTF tables are assembled separately, the definition of additional symbolic linkages in the form of EXTRN-ENTRY symbols will be the programmer's responsibility.

Appendix C contains a full description of the different symbolic linkages that must be defined when separately assembled programs are edited.

When the parameter SEPASMB=YES is specified in a DTF macro instruction, a CATALR card with the file name given to the DTF is generated ahead of the object deck. When the parameter SEPASMB=YES is specified in an xxMOD macro instruction, a CATALR card with the module name is generated ahead of the object deck.

Cataloging DTF tables and logic modules to the relocatable library is recommended to lessen user coding effort and to minimize total time needed to prepare and test programs using logical IOCS. The use of DTF tables cataloged in the relocatable library requires standardization of the labels referred to by the DTF's, so that these tables may be used by different programs.

If the I/O modules, DTF tables, and the main program are assembled together, the linkage editor will search the input stream and will resolve the symbolic linkages between tables and I/O modules by using the External Reference information (V-type address constants generated in DTF tables) and the Section Definition information (CSECT definitions in logic modules). Further information may be found in the Linkage Editor section, under Structure of a Program, in the System Control and System Service Programs publication listed in the Abstract.

If any of the elements that constitute a program are assembled separately, the different "object modules" (assemblies) may be supplied to the input stream at linkage-edit time, and the linkage editor will resolve the symbolic linkages between them.

If any of the separately assembled elements has been cataloged to the relocatable library, the linkage editor will find unresolved external references in the input stream and will perform the AUTOLINK function, searching the relocatable library for a relocatable module whose name matches identically the external reference. If the module is not defined in the relocatable library, the external references to this

name will be unresolved. Therefore, if the modules are assembled separately and cataloged to the relocatable library, the programmer must determine that at least one of the DTF's in his program includes a module name that can be successfully AUTOLINK'ed from the relocatable library.

Programmer control of the module to be AUTOLINK'ed from the relocatable library is achieved by using the MODNAME parameter in the DTF macro instruction. This overrides the standard module name generated by the macro definition.

LINKAGE-EDITING PREASSEMBLED LOGIC MODULES

A small number of IOCS logic modules can serve a large number of DTF macros. (This is applicable only to CDMOD, PRMOD, SDMOD, ISMOD, DAMOD, PTMOD, and MTMOD.) For example, the module shown below can serve the 64 different DTFMT files possible using the six following options: TYPEFLE=INPUT or OUTPUT, RECFORM=FIXUNB or FIXBLK, WORKA=YES or NO, IOAREA2=Name or (not used), CKPTREC=YES or NO, and READ=FORWARD or BACK.

```
                                Col 72

MTMOD                            X
      TYPEFLE=INPUT,             X
      RECFORM=FIXUNB,            X
      WORKA=YES,                 X
      CKPTREC=YES,               X
      READ=BACK
```

The same module also serves files with varying block sizes, record sizes, I/O area addresses, and exit addresses.

A preassembled logic module may be furnished to the linkage editor in three ways:

1. INCLUDE the module from SYSIPT.
2. INCLUDE the module from the system relocatable library.
3. AUTOLINK the module from the system relocatable library.

If a module is included from SYSIPT, its name offers no problem. The user assumes responsibility for its name and functional "match" to the DTF's in his program.

If a module is INCLUDEd from the system relocatable library, the situation is similar. The user should assure himself that the desired modules have already been cataloged to the library by consulting a DSERV listing of the library.

If the module is to be AUTOLINKed from the system relocatable library, the user must determine whether the module name generated by the DTF (or furnished by the MODNAME parameter) coincides exactly with the name in the system relocatable library. If the names are identical, the AUTOLINK is accomplished. Otherwise, the user must either INCLUDE some module that meets these needs (from SYSIPT or from the system relocatable library) or consider the logic module needs of other DTF's in his program. This latter technique is discussed in the following paragraph.

If a needed module is not available in the relocatable library, the user should determine if any other DTF will need a module which (a) is named in the library and (b) furnishes at least the functions required by the first DTF. For example, the following DTF generates a request for the module named IJFFZZZZ.

```
                                Col 72

FILE1 DTFMT                      X
            TYPEFLE=INPUT,       X
            RECFORM=FIXUNB,      X
            IOAREA1=A1,          X
            IOAREA2=A2
```

The following DTF generates a request for the module named IJFFZZWZ.

```
                                Col 72

FILE2 DTFMT                      X
            TYPEFLE=OUTPUT,      X
            RECFORM=FIXBLK,      X
            IOAREA1=A3,          X
            WORKA=YES
```

If the module named IJFFZZWZ is available in the system relocatable library and if the two files are defined in the same assembly, the AUTOLINK facility can be used without further problem. However, if only the first file is defined in the assembly and if no other IJFxxxxx modules are cataloged in the system relocatable library, the user should either furnish a private copy of the IJFFZZZZ module at linkage-edit time or INCLUDE the larger module (IJFFZZWZ). In systems with ample main storage and/or for small programs, the user may choose to sacrifice a modest amount of main storage to achieve simplicity in linkage editing.

An installation could conceivably furnish every possible IOCS module in the system library--several hundred in all. This

technique would severely restrict the capabilities and performance of the system; it is not recommended. Instead, each installation should generate a certain set of IOCS logic modules when the system residence is built, based on equipment, installation standards for record formats and exits, etc. Any user requiring a special tailored module can generate it by using the logic module macro instruction and its specific parameters.

To resolve the symbolic linkages between a superset module, and DTF tables that can be processed by this module but whose name does not match identically the name of the logic module (because the DTF does not use all the functions provided by the module), ENTRY points are generated in addition to the CSECT in the logic module. These ENTRY points will define all the subset module names that can be handled by the superset module. V-type address constants may then be resolved against the ENTRY point if they do not match the CSECT name. For example, the module named IJFFZZWZ has a secondary entry point named IJFFZZZZ. This explains why AUTOLINK will work in the previous example. However, AUTOLINK can be used successfully only with catalog names (which correspond to the CSECT name) but not with secondary entry points.

If the programmer gives an explicit module name to the xxMOD macro-instruction, this name will override the standard module name in the CSECT definition, and no ENTRY points will be generated. The DTF tables that will access the module may employ the parameter MODNAME=Name to link to the previously named logic module.

MACRO-INSTRUCTION FORMAT

Macro instructions have the same format as assembler statements. That is, each macro instruction can consist of a name field, an operation field, and an operand field.

The name field in the macro instruction may contain a symbolic name. Some macro instructions require a name, e.g., CCB, TECB, DTFxx.

The operation field must contain the mnemonic operation code of the macro instruction.

The parameters in the operand field must be written in one of three formats: positional, keyword, or mixed.

POSITIONAL OPERANDS: In this format the parameter values must be in the exact order shown in the macro instruction discussion. Each parameter, except the last, must be followed by a comma with no imbedded blanks. If a parameter is to be omitted in the macro instruction and following parameters are included, a comma must be inserted to indicate the omission. No commas need to be included after the last parameter. Column 72 must contain a continuation punch if the parameters fill the operand field and overflow into another card. Any nonblank character in column 72 causes the next parameter to be read.

For example, GET uses the positional format. A GET for a file named CDFILE using WORK as a work area is punched:

```
GET  CDFILE,WORK
```

KEYWORD OPERANDS: The exact parameters used are equated to a keyword value. Thus, a parameter written in keyword format has the form:

```
LABADDR=MYLABELS
```

where LABADDR is the keyword and MYLABELS is the parameter. The association of parameters is performed through the use of keywords. The parameters in the macro instruction may appear in any position, and any that are not required may be omitted. Different keyword parameters may be punched in the same card, each followed by a comma, like the positional type. Or, they may be punched in separate cards as in Figure 4.

MIXED FORMAT: The parameter list contains both positional and keyword operands. The keyword operands can be written in any order, but they must be written to the right of any positional operands in the macro instruction.

ENTRY CARDS FOR DECLARATIVE MACROS

The parameters of the DTFxx and the module generation macro instructions can be punched in a set of entry cards in the assembler format previously described. An example of the entry cards used for a DTFMT macro instruction is shown in Figure 4. The macros may be assembled in any order.

The first entry card is a header card, and the continuation cards are detail cards. The header card is punched with:

- The symbolic name of the file in the name field. Programming Note: Avoid defining symbols beginning with IJ; they may conflict with IOCS symbols beginning with IJ. Avoid symbols which are identical to a filename plus a single character suffix. For example, for the filename RECIN, IOCS generates the symbols RECINS, RECINL, etc by concatenating the filename with an additional character.

In a DTF, the symbolic filename may be up to seven characters long. If the Job Control VOL card is required for the file defined by the DTF, the filename must be the same as the filename used on the VOL card.

For a module generation macro, the name may or may not be specified. See Generation of Module Names in DTF Tables and Logic Modules in this publication.

- The macro instruction mnemonic operation code in the operation field.
- Keyword entries in the operand field, if desired.
- A continuation punch in column 72, if detail cards are necessary.

The detail cards follow the header card, and they may be arranged in any order. Each detail card is blank in the name and operation fields and is punched beginning in column 16, with one or more keyword operands, separated by commas. All detail cards except the final one must be punched with a comma immediately following the last operand and with a continuation punch in column 72. They may contain comments if a space is left after the comma following the last operand.

MACRO-INSTRUCTION CONVENTIONS

The conventions used in this publication to illustrate macro instructions are as follows:

1. Upper-case letters and punctuation marks (except as described in items 3 and 4 below) represent information that must be coded exactly as shown.
2. Lower case letters and terms represent information that must be supplied by the programmer.
3. Information that is contained within brackets [ ] represents an option that can be included or omitted, depending on the requirements of the program.
4. An ellipsis (a series of three periods enclosed by commas) indicates that a variable number of items may be included.
5. Options contained within braces { } represent alternatives, one of which must be chosen.
6. [Name / Label / Address] — A name-field symbol in this assembly, or an operand of an EXTRN statement, or * (the location counter).
7. Filename — Symbol appearing in the name field of a DTF macro instruction.
8. [n / (r)] — Self-defining value, such as X'04', (15), B'010'.
9. Length — Absolute expression, as defined in the Assembler publication.
10. [A / B / C] — Underlined elements represent an assumed option in the event a parameter is omitted.
11. {Name / (r)} — Ordinary Register Notation
12. {Name / (0)} {Name / (1)} — Special Register Notation (Ordinary Register Notation can be used).

Register Notation

Certain operands can be specified in either of two ways. One, the user can specify the operand directly. Or two, he can preload the address of the value into a register before executing the macro instruction and specify the register in the macro instruction. (The registers that can be used are discussed under Register Usage in this publication.) The second method is known as (ordinary) register notation. When the macro instruction is assembled, instructions are generated to pass the information specified in the operand to IOCS or the Supervisor.

For example, if an operand is written as (8), and if the corresponding parameter is to be passed to the Supervisor in register 0, the macro expansion will contain the instruction LR 0,8.

The user can save main storage and execution time by using what is known as special register notation: writing the operand as either (0) or (1).

This notation is special for two reasons:

- The register notation designation of registers 0 and 1 is not allowed unless specifically designated.

- The designation must be made by the specific three characters (0) or (1), rather than by the general form of an absolute expression enclosed in parentheses. For example, even though the absolute expression REG could be equated to 0, (REG) must not be written instead of (0) when special register notation is intended. If this were done, the macro expansion would contain a useless LR 0,REG instruction.

The format description of each macro instruction shows whether special register notation can be used, and for which operands. For example,

$$\text{GET} \quad \left\{ \begin{matrix} \text{Filename} \\ \\ (1) \end{matrix} \right\} \left[, \left\{ \begin{matrix} \text{Workname} \\ \\ (0) \end{matrix} \right\} \right]$$

The format description shows that the filename operand can be written as (1), and the workarea operand as (0). If either of these special register notations is used, the user's problem program must have loaded the designated parameter register before execution of the macro expansion. Ordinary register notation could also have been used.

Imperative macro instructions are issued by the programmer and initiate such functions as opening a file, making records available for processing, writing records that have been processed, etc. The macro instructions provided by IBM for input/output control are present in this section in the following groups.

- Initialization: OPEN, OPENR, and LBRET
- Processing Records Sequentially: GET, PUT, RELSE, PRTOV, CNTRL, TRUNC, RESCN, DSPLY, RDLNE, and WAITF
- Processing Work Files: READ, WRITE, CHECK, NOTE, POINTR, POINTW, and POINTS
- Processing DASD Records by the Direct Access Method: READ, WRITE, WAITF, and CNTRL
- Processing Direct Access Storage Device (DASD) Records by the Indexed Sequential File Management System: SETFL, ENDFL, WRITE, READ, WAITF, SETL, ESETL, GET, and PUT
- Completion: CLOSE, CLOSER, LBRET, FEOV, and SEOV

Figure 5 summarizes when these macros are used.

When necessary, the detail entries of the DTF macro associated with the file are cross-referenced in Figure 5. For example: **FILABL=STD.**

INITIALIZATION

Before processing a file, the file is usually readied for use by issuing an OPEN macro. Exception, OPEN is not used with DTFCN files. OPEN optionally checks or writes standard labels. The user can exit from the IOCS OPEN routine to write or check user labels or nonstandard labels. When opening a file to be processed by physical IOCS or direct access method, the user can exit to process his XTENT information. The LBRET macro is issued in user routines to return to IOCS.

Information on labels is contained in the Data Management Concepts publication listed on the cover and in Appendix A of this publication.

OPEN MACRO

Op	Operand
for programs which are not self-relocating	
OPEN	{Filename1 / (r1)} [,{Filename2 / (r2)}...,{Filenamen / (rn)}]
for self-relocating programs	
OPENR	{Filename1 / (r1)} [,{Filename2 / (r2)}...,{Filenamen / (rn)}]

The OPEN macro instruction is used to activate all files that are processed with LIOCS except printer-keyboard files, and certain files that are processed with PIOCS (see the discussion on PIOCS). Files that have been CLOSEd cannot be OPENed within the same job step.

When the operation OPEN is used, the symbolic address constants which OPEN generates from the parameter list are not self-relocating. When OPENR is specified, the symbolic address constants are self-relocating.

Self-relocating programs using LIOCS must use the OPENR macro-instruction to activate all files, including printer-keyboard files. The OPENR macro, in addition to activating files for processing, relocates all address constants within the DTF tables specified in the operand field.

The symbolic name of the file (DTF filename) is entered in the operand field. A maximum of 16 files may be opened with one OPEN by entering the filenames as additional operands. Alternately, the user can load the address of the DTF filename in a register and specify the register using ordinary register notation. Filename should not be preloaded into Register 0.

Whenever an input/output DASD or magnetic tape file is to be opened and the user plans to process user-standard labels (UHL or UTL), or nonstandard tape labels, he must provide the information for checking or building the labels. If this information is obtained from another input file, that file must be opened, if necessary, ahead of the DASD or tape file. This is done by specifying the input file ahead of the tape

or DASD file in the same OPEN, or by issuing a separate OPEN preceding the OPEN for the file.

Opening Other Files

For the card reader, card punch, paper tape reader, and printer, OPEN simply makes the file available for input or output.

For 1403 printers with the Universal Character Set feature, data checks are suppressed unless the user specifies UCS=OFF in the DTFPR for the file.

When logical IOCS is used with the IBM 1285 Optical Reader or IBM 1287 Optical Reader, the OPEN macro must be issued at the beginning of each input roll when processing journal tapes. When processing documents, OPEN must be issued to make the file available. OPEN allows header (identifying) information to be entered at the 1285 or 1287 keyboard, if desired, for journal tape or cut documents (1287). When header information is entered, it is always read into IOAREA1, which must be large enough to accommodate the desired header information.

MACRO INSTRUCTION	TYPE OF PROCESSING WITH LOGICAL IOCS															
	Sequential											Indexed Sequential System				
	2311 Disk Drive	2400 Series Magnetic Tape Unit	1442/2501/2520/2540 Reader	1442/2520/2540 Punch	1403/1404/1443/1445 Printer	1052 Printer-Keyboard	2671 Paper Tape Reader	2321 Data Cell	1285 Optical Reader	1287 Optical Reader	Direct Access Method	Load File	Add Records	Random Retrieve	Sequential Retrieve	PHYSICAL IOCS
Initialize																
OPEN	X	X	X	X	X		X	X	X	X	X	X	X	X	X	X[1]
LBRET[2]	X	X						X			X					X
Process																
Transfer Records																
GET[10]	X	X	X[3]			X	X	X	X	X					X	
PUT	X[4]	X	X[5]	X	X	X		X[4]							X[12]	
READ[11]										X	X			X		
WRITE											X	X	X	X[12]		
RELSE[6]	X	X				X										
TRUNC[7]	X	X				X										
WAITF[11]										X	X	X		X		
EXCP																X
WAIT																X
RDLNE[10]									X	X						
RESCN[11]										X						
DSPLY[11]										X						
Set Mode																
SETFL												X				
ENDFL												X				
SETL															X	
ESETL															X	
Non-Data Operations																
CNTRL	X	X	X	X	X			X	X	X	X					
CHNG[8]		X														X
PRTOV					X											
CCB																X
Work Files for DASD and Magnetic Tape																
READ	X	X														
WRITE	X	X														
CHECK	X	X														
NOTE	X	X														
POINTR	X	X														
POINTW	X	X														
POINTS	X	X														
Complete																
CLOSE	X	X	X	X	X		X	X	X	X	X	X	X	X	X	X[1]
LBRET[2]	X	X						X								X
FEOV		X														X[9]

Notes:
1. Required only for DASD or tape files with standard labels for DTFPH and for file protected DASD files.
2. Applies only if DTFSR, DTFMT, DTFDA, DTFSD, or DTFPH LABADDR is specified or if DTFPH or DTFDA XTNTXIT is specified.
3. In the 2540, GET normally reads cards in the read feed. If TYPEFLE=CMBND is specified, GET reads cards at the punch-feed-read station.
4. PUT rewrites an input DASD record if UPDATE is specified.
5. In the 1442, 2520, or 2540, PUT punches an input card with additional information if TYPEFLE=CMBND is specified.
6. Applies only to blocked input records.
7. Applies only to blocked output records.
8. Applies only when two selector channels and one or more 2-channel, simultaneous-read-while-write tape control units are installed.
9. Applies only to output tape files with standard labels.
10. For 1287, applies only to journal tape processing.
11. For 1287, applies only to document processing.
12. Rewrites an input record.

Figure 5. Macro Instructions for Input/Output Control

OPENING DASD FILES

When a DASD file is processed, the user must supply the following Job Control cards: a VOL and DLAB card for each logical file to be processed and an XTENT card for each separate area (or extent as it is more commonly called). OPEN uses the information supplied in these cards and also certain information from the DTF table for the file to process labels.

The extent(s) for a file must either coincide with or be within an existing extent(s) that is defined in the Volume Table of Contents (VTOC). That is, on input, IOCS will open only an existing file or a subset of an existing file. On output, the file to be written cannot overlap existing unexpired files; IOCS will not destroy an unexpired file without an explicit request from the user.

SEQUENTIAL PROCESSING--OUTPUT

When a multivolume DASD file is to be created using sequential processing, only one extent is processed at a time, thus only one pack need be mounted at a time.

When a file is opened, OPEN checks the standard VOL1 label and checks the extents specified in the XTENT cards:

1. the extents must not overlap each other,
2. the first extent must be at least two tracks long if user standard labels are to be created,
3. Only extent types 1 and 128 are valid.

The data extents of a sequential DASD file can be type 1, type 128, or both. Type 128 extents are called split cylinder extents and use only a portion of each cylinder in the extent. The portion of the cylinder used must be within the head limits of the cylinder and within the range of the defined extent limits. For example, 2 files can share 3 cylinders--one file occupying the first 2 tracks of each cylinder and the other file occupying the remaining tracks. In some applications, the use of split cylinder files reduces the access time.

Then, OPEN checks all the labels in the VTOC to ensure that the file to be created will not destroy an existing file whose expiration date is still pending and that the extents specified in the XTENT cards do not overlap existing extents. After the VTOC checks, OPEN creates the standard label(s) for the file and writes the label(s) in the VTOC.

If the user wishes to create his own user standard (UHL or UTL) labels for the file, he must include the DTF entry LABADDR. OPEN reserves the first track of the first extent for the user header and trailer labels. Then the user's label routine is given control at the address specified in LABADDR.

After the header labels are built, the first extent of the file is ready to be used. The extents are made available in the order of the sequence numbers on the XTENT cards. When the last extent on the mounted volume has been filled, user standard trailer labels can be built. Then the next volume that was specified in the XTENT cards is mounted and opened in the above manner.

For a file-protected DASD, when OPEN makes the first extent of the new volume available, it makes the extent(s) from the previous volume unavailable.

When the last extent on the last volume of the file has been processed, OPEN issues a message. The user has the option of canceling the job or typing in an XTENT on the Printer-Keyboard and continuing the job.

SEQUENTIAL PROCESSING--INPUT

In a multivolume file (a file having extents on more than one disk pack), only one extent is processed at a time, and thus only one pack need be mounted at a time.

When a volume is opened, OPEN checks the standard VOL1 label and goes to the VTOC to check the file label(s). OPEN checks the extents specified in the XTENT cards against the extents in the labels to make sure the extents exist. If LABADDR is specified, OPEN makes the user standard header (UHL) labels available, one at a time, to the user for checking.

After this, the first extent of the file is ready to be processed. The extents are made available in the order of the sequence number on the XTENT cards. Note the same XTENT cards that were used to build the file can be used when the file is used as input. When the last extent on the mounted volume has been processed, the user standard trailer labels are made available for checking one at a time. The next volume is opened in the above manner.

For DASD devices that are file protected when OPEN makes the first extent of the new volume available, OPEN makes the extent(s) from the previous volume unavailable.

DIRECT ACCESS PROCESSING--OUTPUT

If a file is to be created using the direct access method of processing, all volumes used must be mounted at the same time, and all the volumes are opened before the processing is begun.

For each volume, OPEN checks the standard VOL1 label and checks the extents specified in the XTENT cards:

1. The extents must not overlap;

2. Only type-1 extents can be used;

3. If user standard header labels are to be created, the first extent must be at least two tracks long.

Then OPEN checks all the labels in the VTOC to ensure that the file to be created will not destroy an existing file while the expiration date is still pending. After the VTOC check, OPEN creates the standard label(s) for the file and writes the label(s) in the VTOC.

If the user wishes to create his own user labels (UHL) for the file, he must include the DTF entry LABADDR. OPEN reserves the first track of the first extent for these header labels and gives control to the user's label routine.

If the XTNXIT entry is specified, OPEN stores the address of a 14-byte extent information area in register 1. (See DTFDA for the format of this area.) Then OPEN gives control to the user's extent routine. The user can save this information for use in specifying record addresses.

After the user labels have been written, the next volume is opened. When all the volumes have been opened, the file is ready for processing. If the DASD device is file protected, all extents specified in XTENT cards are available to the user.

DIRECT ACCESS PROCESSING--INPUT

Direct access processing requires that all volumes containing the file be on-line and ready at the same time. All volumes used are opened before any processing can be done.

For each volume, OPEN checks the standard VOL1 label and then checks the file label(s) in the VTOC. OPEN checks some of the information specified in the XTENT cards for that volume. If LABADDR is specified, OPEN makes the user standard header labels available one at a time for checking.

If the XTNTXIT entry is specified, OPEN stores the address of a 14-byte extent information area in register 1. (See DTFDA for the format of this area.) Then OPEN gives control to the user's extent routine. The user can save this information for use in specifying record addresses. Then the next volume is opened. After all the volumes have been opened, the file is ready for processing. If the DASD device is file protected, all extents specified in XTENT cards are available for writing.

INDEX SEQUENTIAL PROCESSING--OUTPUT

When a file is to be created or extended using index sequential processing, those volumes of the file that will be written on are opened as output files. If the file consists of more than one volume, all the volumes must be on line and ready when the file is first opened.

For each volume, OPEN checks the standard VOL1 label and performs extensive checks on the extents specified in the XTENT cards for that volume. The extents must meet the following conditions:

1. All prime data extents must be continuous;

2. The master and cylinder index extents must be continuous and on the same unit;

3. No extents must overlap;

4. Only type 1, 2, or 4 extents are valid.

5. The extent sequence numbers must be in the following order: 0 for master index, when present; 1 for cylinder index; 2, 3, 4,... for the prime data and independent overflow tracks. The XTENT cards for the independent overflow tracks can be placed either before or after all the XTENT cards for the prime data extents.

OPEN checks all the labels in the VTOC to ensure that the file to be created will not destroy an existing file while the expiration date is pending. Any expired labels are deleted from the VTOC. After the VTOC check, OPEN creates the standard labels for the file and writes the labels in the VTOC. If the DASD device is file protected, all extents specified in the XTENT cards are available for writing.

INDEX SEQUENTIAL PROCESSING--INPUT

All volumes containing an index sequential file must be on-line and ready when the file is first opened.

For each volume, OPEN checks the extents specified in the XTENT cards for that volume (for example, checks that the data extents are continuous). OPEN also checks the standard VOL1 label and then goes to the VTOC to check the file label(s). Then the next volume is opened. After all the volumes have been opened, the file is ready for processing. If the DASD device is file protected, all extents specified in XTENT cards are available to the user.

PIOCS--SINGLE VOLUME MOUNTED--OUTPUT

When processing with physical IOCS, OPEN is used only if the user wants to build standard labels. When the first OPEN for the volume is issued, OPEN checks the standard VOL1 label and the extents specified in the XTENT cards for the mounted volume:

1. The extents must not overlap each other;

2. If user standard header labels are to be written, the first extent must be at least two tracks;

3. Only types 1 and 128 extents are valid.

Then, OPEN checks all the labels in the VTOC to ensure that the file to be created will not destroy an existing file whose expiration date is still pending.

If the user wishes to create his own user standard header (UHL) labels for the file, he must include the DTF entry LABADDR. OPEN reserves the first track of the first extent for these labels and gives control to the user's label routine.

After this, the first extent of the file is ready to be used. Each time the user determines that he has completed all processing for an extent, he issues another OPEN for the file and that OPEN makes the next extent available. When the last extent on the last volume of the file has been processed, OPEN issues a message. The user has the option of canceling the job, or typing in an XTENT on the Printer-Keyboard and continuing the job.

If the system provides DASD file protection, only the extents that are opened for the mounted volume are available to the user.

PIOCS--SINGLE VOLUME MOUNTED--INPUT

When processing with physical IOCS, OPEN is used only if the user wants to check standard labels.

When the volume that is mounted is opened for the first time, OPEN checks the extents specified in the XTENT cards (for example, checks that the extent limit address for the device being opened is valid). OPEN also checks the standard VOL1 label and then checks the file label(s) in the VTOC.

If LABADDR is specified, OPEN makes the user standard labels (UHL) available to the user one at a time for checking. Then OPEN makes the first extent available for processing.

Each time the user determines that he has completed all processing for an extent, he issues another OPEN for the file and OPEN makes the next extent available. If another extent is not available, OPEN stores the character 'F' (for EOF) in byte 31 of the DTFPH table. The user can determine the end of file by addressing and checking the byte at Filename+30.

If the system provides DASD file protection, only the extents that are opened for the mounted volume are available to the user.

PIOCS--ALL VOLUMES MOUNTED--OUTPUT

If all the volumes to be used are mounted when creating an output file with physical IOCS, all the volumes are opened before the file is processed. OPEN is used only if standard labels are to be checked.

For each volume, OPEN checks the standard VOL1 label and checks the extents specified in the XTENT cards:

1. The extents must not overlap each other;

2. Only type-1 extents can be used;

3. If user standard header labels are to be created, the first extent must be at least two tracks long.

Then OPEN checks all the labels in the VTOC to ensure that the file to be created will not destroy an existing file while the expiration date is still pending. After this check, OPEN creates the standard label(s) for the file and writes the label(s) in the VTOC.

If the user wishes to create his own user standard header (UHL) labels for the file, he must include the DTF entry LABADDR. OPEN reserves the first track of the first extent for these labels and gives control to the user's label routine.

If the XTNTXIT entry is specified, OPEN stores the address of a 14-byte extent information area in register 1. (See Processing Records with Physical IOCS: DTFPH for the format of this area.) Then OPEN gives control to the user's extent routine. The user can save this information for use in specifying record addresses. If the user's DASD file is file protected, he cannot write on any extents while in his XTNTXIT routine.

When the checking is complete, the user returns control to OPEN by issuing the LBRET 2 macro. Then the next volume is opened. After all the volumes have been opened, the file is ready for processing.

PIOCS--ALL VOLUMES MOUNTED--INPUT

When all volumes containing the file are on-line and ready at the same time, the volumes are opened one at a time before any processing is done. OPEN is used only when standard labels are to be processed.

For each volume, OPEN checks the extents specified in the XTENT cards, and checks the standard VOL1 label on track 0 and the file label(s) in the VTOC.

If LABADDR is specified, OPEN makes the user standard labels available, one at a time, for checking.

If XTNTXIT is specified, OPEN stores the address of a 14-byte extent-information-area into register 1. (See Processing Records with Physical IOCS: DTFPH for the format of this area.) Then OPEN gives control to the user's extent routine. For example, the user can save this area and use the information to provide a method of specifying the address of the record to be processed. If the DASD file is file protected, the user cannot write on any extents while in his XTNTXIT routine.

Each volume is completely opened before the next volume is opened. When all volumes are opened the file is ready for processing.

VTOC CHECKING FOR OUTPUT FILES

When an output file is opened, OPEN checks the Volume Table of Contents (VTOC) to determine whether the output file can be written on the volume being opened. If OPEN determines that the output file will overlay an existing file whose expiration date has expired, OPEN deletes the expired label(s) from the VTOC. This in effect removes the file from the volume. In a multi-volume file, the file may be removed from all the volumes that it occupies or from only some of the volumes.

If OPEN determines that the expiration date of an existing file to be overlaid by the output file has not expired, the old file cannot be destroyed automatically. The user has the following choices.

For sequential or physical IOCS processing:

1. Terminate the job.
2. Bypass the extent. If more extents have been specified, the next one will be checked and supplied if it is available. If no more extents are specified, a message will be issued and the user may type in an extent from the 1052, or he may terminate the job.
3. Delete the unexpired file.

For work file and direct access processing:

1. Terminate the job.
2. Bypass the extent. If more extents have been specified, the next one will be checked and supplied if available. If no more extents are available, the job will be terminated.
3. Delete the unexpired file.

For index sequential processing:

1. Terminate the job.
2. Delete the unexpired file.

WRITING DASD USER STANDARD LABELS

When user standard trailer (UTL) and/or header (UHL) labels are to be written, the user must specify the DTF entry LABADDR. This causes OPEN to reserve the first track of the first data extent for the user label area. User labels cannot be created for a file whose first extent is a split cylinder extent or for an ISFMS file. When LABADDR is specified, at least one user header label and one user trailer label must be written.

IOCS uses bytes 1-4 of the 80-byte label for the label identification (for example, UHLx, x=1, 2, ..., 8) and the user can use the other 76 bytes as he wishes. The maximum number of user labels is 8 header and 8 trailer labels for a 2311 file, and 5 of each for a 2321 file.

OPEN loads an alphabetic 'O', 'V', or 'F' in register 0. O indicates header labels; V indicates end-of-volume labels; and F indicates end-of-file labels. The user can test this character to determine whether header, end-of-volume, or end-of-file labels should be written. OPEN also loads the address of an 80-byte IOCS label area in register 1.

The OPEN stores the label identification (UHLx or UTLx) that it generates in bytes 1-4 of the IOCS label area. The user can test the identification to determine the type and the number of the label.

Then OPEN gives the control to the user's label routine at the address specified in LABADDR. While in his label routine, the user cannot issue a macro that calls a transient routine. For example, OPEN, CLOSE, DUMP, PDUMP, CANCEL, and CHKPT cannot be issued. If the user's DASD file is file protected, no extents are available for writing while in the user's label routine.

The user can build his labels in either of the following ways.

1. Build an 80-byte (or a 76-byte) label in the user area of main storage, and load the address of the label area (or label area minus four if a 76-byte label was built) into register 0 before issuing the LBRET macro. (When the label is moved into the IOCS area, IOCS adds four bytes to the address in register 0.)

2. Build a 76-byte label in the IOCS area at the address (that IOCS supplies in register 1) plus four, and load the contents of register 1 to register 0 before issuing the LBRET macro.

Note that the IOCS area of main storage is a part of the Supervisor. If the program is to be executed on a system with the storage protection feature, method 1 must be used because the user cannot write into the Supervisor area. Thus, no user standard label routine using the second method can be executed in a multiprogramming environment.

When the label is ready to be written, the user issues the LBRET macro, which returns control to IOCS. If LBRET 2 is used, OPEN writes the label and returns control to the user's label routine unless the maximum number of labels has been written. If LBRET 1 is used, the label set is terminated; no more labels can be created.

When IOCS receives control, the IOCS routines move the label from the address the user loaded in register 0 into the IOCS label area. If the maximum number of labels has not been written, IOCS increases the identification number by 1 and returns to the user's label routine unless LBRET 1 was used. If the maximum number of labels has been created, IOCS automatically terminates the label set.

CHECKING DASD USER STANDARD LABELS

When a DASD file contains user standard trailer and/or header labels, IOCS makes these labels available one at a time to the user if LABADDR is specified in the DTF for the file. IOCS reads a label and stores information for the user in registers 0 and 1:

register 1 - the address of the label just read,

register 0 - an alphabetic 'O', 'V', or 'F'
O indicates header labels
V indicates end-of-volume labels
F indicates end-of-file labels.

After initializing the registers, IOCS enters the user's routine at the label specified in LABADDR. If the user's DASD file is file protected, no extents are available for writing while the user's routine is being executed.

In his routine, the user can process the label, using logical IOCS macros if desired. The user cannot issue a macro that calls a transient routine. For example, OPEN, CLOSE, DUMP, PDUMP, CANCEL and CHKPT cannot be issued.

If the labels are to be checked against information obtained from another input file, that file must be opened ahead of the DASD file.

When the user is finished with that label, he issues a LBRET 2 macro, which causes OPEN to read the next label. However, if the end-of-file record at the end of the labels is read instead, OPEN automatically terminates the label checking.

If the user wishes to end label checking before all the labels have been read, he issues a LBRET 1 macro.

POSITIONING TAPE FILES

When a magnetic tape file with standard labels (FILABL=STD) is opened, IOCS expects the first record read to be a label. On output this label must be the label identifying the file that will contain the output data. On input, this label can be any label preceding the file; IOCS locates the correct file by the file sequence number. On input the first record is a label if the tape file being opened is the first file on the reel and if IOCS rewinds the reel (see REWIND). If no rewind is specified for either an input or an output file, or if an output file starting in the middle of the reel is opened, it is the user's responsibility to properly position the tape prior to the OPEN. The tape should be positioned immediately past the last tape mark of the preceding file. The user can employ the MTC command of Job Control for this function. If the tape is improperly positioned, IOCS indicates an error condition by issuing a message to the operator.

When a magnetic tape with nonstandard labels is opened, the tape must be positioned to the first label that the user wishes to process. The MTC job control command can be used to skip the necessary number of tapemarks or records to position the file.

On both input and output, an unlabeled file must be positioned at the location to be processed. That is, the tape must be either at load point or at the tape mark following the data for a previous file.

OPENING TAPE OUTPUT FILES

For a magnetic tape output file, OPEN rewinds the tape as specified in the DTFSR or DTFMT entry REWIND. (No rewind is performed if the file is defined by DTFPH.)

Writing Standard Labels

When standard header labels are to be written (STD specified in DTFSR or DTFMT FILABL, or OUTPUT in DTFPH TYPEFLE), the user must supply the Job Control VOL and TPLAB cards. When the OPEN is issued, the volume (VOL1) label is checked and the old file header, if present, is read and checked to make sure that the file on the tape is no longer active and may be destroyed. If the file is inactive or if a tape mark was read, the tape is backspaced and the new file header (HDR1) label is written with the information the user supplies in the Job Control TPLAB card. The volume label is not rewritten.

Writing User Standard Labels

A maximum of eight user standard (UHL) labels can be written following the standard header (HDR1) label. The user standard labels are 80 bytes long and are built entirely by the user. Bytes 1-4 must contain the label identification (UHLx, where x=1,2,...,8); the other 76 bytes can be used as desired.

To write these labels, the user specifies LABADDR in the DTF for the file. When the file is opened, OPEN loads an alphabetic O in the low-order byte of register 0. The user can test for the O, which indicates that the file is being opened and that a header label should be built. IOCS also loads the address of an 80-byte IOCS output area in register 1. Then OPEN gives control to the user's routine at the address specified in LABADDR.

The user can build his labels in either of the following ways:

1. Build the label in the user area of main storage, and load the address of the label into register 0 before issuing the LBRET macro.

2. Build the label in the IOCS area at the address that IOCS supplies in register 1, and load the address of the area from register 1 to register 0 before issuing the LBRET macro.

Note that the IOCS area of main storage is a part of the Supervisor. If the program is to be executed on a system with the storage protection feature, method 1 must be used because the user cannot write into the Supervisor area. Thus, no user standard label routine using the second method can be executed in a multiprogramming environment.

When the label is ready to be written, the user issues the LBRET macro, which returns control to IOCS. LBRET 2 returns control to the user's routine after IOCS writes the label. LBRET 1 must be used to terminate the label set.

When IOCS receives control, IOCS writes the label on the magnetic tape and either returns control to the user (LBRET 2) or writes a tape mark (LBRET 1).

Writing Nonstandard Labels

To write nonstandard labels, the user must specify FILABL=NSTD and LABADDR=name. He must also write his own channel program and

use physical IOCS macros to transfer the labels from main storage onto tape. For an example see Appendix D.

When a file is opened, OPEN supplies the hexadecimal representation of the symbolic unit currently being used in the two low-order bytes of register 1. See Figure 36, Command Control Block, for these values. For example, this information might be useful when one LABADDR routine is used for many files.

IOCS also stores an alphabetic O in the low-order byte of register 0. The user can test for the O, which indicates that the file is being opened and that header labels should be built.

Then the user's routine is given control at the address of the symbol in LABADDR. Physical IOCS macros must be used to transfer the labels from main storage onto tape. A Command Control Block (CCB) and a Channel Command Word(s) (CCW) must be established, and an EXCP macro must be issued for each label record. See Processing Records with Physical IOCS.

Logical IOCS macros can be used for any processing other than transferring the labels from main storage to tape. If any logical IOCS macros, other than LBRET, are used to process nonstandard labels on multi-volume files, the user must save and restore register 15 in his routine.

After all labels have been written, the user returns control to OPEN by use of the LBRET 2 macro. OPEN either writes, or does not write, a tapemark as the user specifies in the DTF for the file. (See the DTF entry TPMARK.) The file is then ready for processing.

Unlabeled Output

OPEN either writes, or does not write, a tapemark as the user specifies in the DTF for the file. (See the DTF entry TPMARK.) The file is then ready for processing.

OPENING TAPE INPUT FILES

When an input file is recorded on magnetic tape, OPEN rewinds the tape according to the specifications in the DTFSR or DTFMT entry REWIND. (No rewind is performed if the file is defined by DTFPH.)

Checking Standard Labels

Both the volume (VOL1) and file header (HDR1) labels are automatically read and checked if standard label checking is specified (STD specified in FILABL, or INPUT specified in the DTFPH parameter TYPEFLE) and if the tape is read forward (FORWARD specified in DTFMT or DTFSR READ). The labels are checked with the information the user supplies to Job Control in the VOL and TPLAB cards.

When the tapemark at the end of the labels (standard or user standard, if present) is read, IOCS opens the next file specified in the OPEN macro, or returns control to the problem program if all files have been opened. The file is then ready for processing.

READING BACKWARDS: If a magnetic tape file that will be read backwards (READ=BACK) is opened, the file trailer label is automatically read and checked if label checking is specified. (The volume label is not repeated at the end of the tape.) Because the file trailer label is processed at this time, it must be complete and contain both the trailer and header information (except HDR) to identify the file. If the file labels were originally written by IOCS routines, the trailer label will be complete. When physical IOCS macros are used to read records backwards, labels cannot be checked (DTFPH must not be specified).

BYPASSING STANDARD LABELS: If an input tape contains standard labels but the user does not want IOCS to check them, FILABL=NSTD should be specified in the file definition. No LABADDR is specified. However, a tapemark must be present immediately following the label set. If user labels exist but the user does not specify LABADDR the user labels are bypassed by IOCS. A tapemark must be present after the last user label.

Checking User Standard Labels

If a magnetic tape contains user standard header labels (UHL1-UHL8) following the standard file header label [or user standard trailer labels (UTL1-UTL8) preceding the standard trailer when reading backwards], the programmer can check them in his own routine. If the information for checking is obtained from another file, that file must be opened ahead of the tape file containing the labels.

OPEN reads a user label, stores the address of the label in register 1, and gives control to the user's routine for checking. The entry point of the routine is specified in the DTF entry LABADDR. If any logical IOCS macros, other than LBRET, are used to process user labels on a multi-volume file, the user must save and restore register 15 in his label routine.

After checking the label, the user must return to the OPEN routine by use of the LBRET macro. If the user wants to process the next label, he uses LBRET 2, and OPEN supplies his routine with the next user label. To bypass the rest of the labels he uses LBRET 1.

Checking Nonstandard Labels

To process nonstandard labels, the user must specify FILABL=NSTD and LABADDR=name. He must also write his own channel program and use physical IOCS macros to read the labels from the tape into main storage. (To bypass nonstandard labels, the user must specify FILABL=NSTD and omit LABADDR. A tapemark must follow the last label.) For an example see Appendix D.

When a file is opened, IOCS supplies the hexadecimal representation of the symbolic unit currently being used in the two low-order bytes of register 1. See Figure 36, Command Control Block, for these values. For example, this information might be useful when one LABADDR routine is used for many files.

Then the user's routine is given control at the address of the symbol in LABADDR. Note that OPEN has not moved the tape. Physical IOCS macros must be used to transfer the labels from tape to main storage. Therefore, the user must establish a Command Control Block (CCB) and a Channel Command Word(s) (CCW). The macro EXCP is used to initiate the transfer. See Processing Records with Physical IOCS.

Logical IOCS macros can be used for any processing other than transferring the labels from tape to main storage. If any logical IOCS macros, other than LBRET, are used when processing multivolume files, the user must save and restore register 15 in his routine.

If nonstandard trailer labels are to be processed on a file that is read backwards, the user must position the tape between the two tapemarks that are at the end of the file before opening the file.

After all labels have been checked, the user returns control to OPEN by use of the LBRET 2 macro. If a tapemark follows the last label, the user can read it or not as desired with no effect on OPEN.

Unlabeled Input

FILELABL=NO must be specified for an unlabeled tape. When OPEN is issued for such a file, OPEN reads a record. If the record is a tapemark, the file is considered open. If a tapemark is not found, OPEN assumes the record read is data and backspaces to the load point.

LBRET MACRO

Name	Operation	Operand
[name]	LBRET	{1 / 2}

The LBRET macro is issued in user subroutines when the user has completed processing and wishes to return control to IOCS. LBRET applies to subroutines that write or check DASD or magnetic tape user standard labels, write or check tape nonstandard labels, or check DASD extents. The operand used depends on the function to be performed.

CHECKING USER STANDARD DASD LABELS: IOCS passes the labels to the user one at a time until the maximum allowable number have been read, or until the user signifies he wants no more. In his label routine, the user issues LBRET 2 if he wants IOCS to read and pass him the next label. If an end-of-file record is read when LBRET 2 is used, label checking is automatically ended. If the user wants to eliminate the checking of one or more remaining labels, he uses LBRET 1.

WRITING USER STANDARD DASD LABELS: The user builds the labels one at a time and uses LBRET to return to IOCS, which writes the labels. LBRET 2 is used if the user wants to have control returned to him after IOCS writes the label. If, however, IOCS determines that the maximum number of labels has been written, label processing will be terminated. LBRET 1 is used if the user wishes to stop writing labels before the maximum number has been written.

CHECKING DASD EXTENTS: When using physical IOCS on an input file with all volumes mounted or when processing with the direct access method, the user can process his extent information. When he finishes all the checking, he returns control to IOCS by using the LBRET 2 macro.

CHECKING USER STANDARD TAPE LABELS: IOCS reads and passes the labels to the user one at a time until a tapemark is read, or until the user signifies he does not want any more labels. LBRET 2 is used if the user wants to process the next label. If IOCS reads a tapemark, label processing is automatically terminated. LBRET 1 is used if the user wants to bypass any remaining labels.

WRITING USER STANDARD TAPE LABELS: The user builds the labels one at a time and returns to IOCS, which writes the labels. When LBRET 2 is used, IOCS returns control to the user (at the address specified in LABADDR) after writing the label. LBRET 1 must be used to terminate the label set.

WRITING OR CHECKING NONSTANDARD TAPE LABELS: The user must process all his nonstandard labels at once. LBRET 2 is used after all label processing is completed and the user wants to return control to IOCS. For an example see Appendix D.

MACROS FOR SEQUENTIAL PROCESSING

The sequential processing macro instructions permit the programmer to store and retrieve records without coding blocking/deblocking routines. The programmer can, therefore, concentrate on processing his data. Another major feature of these macro instructions is the ability to use one or two I/O areas and to process records in either a work area or an I/O area.

The sequential processing routines are designed to provide for overlapping the physical transfer of data with processing. The amount of overlapping actually achieved is governed by the problem program through the assignment of I/O areas and work areas. An I/O area is that area of main storage to or from which a block of data is physically transferred by the logical IOCS. A work area is an area used for processing an individual logical record from the block of data. A work area cannot be used with paper tape records. The I/O area(s) is specified in the associated DTF macro, while the work area is specified in the sequential processing macro.

The following combinations of I/O areas and work areas are possible:

1. One I/O area with no work area
2. One I/O area with a work area
3. Two I/O areas with no work area
4. Two I/O areas with a work area

GET MACRO

Name	Operation	Operand
[name]	GET	{Filename / (1)} [,{Workname / (0)}]

GET makes the next sequential logical record from an input file available for processing in either an input area or a specified work area. It is used for any input file in the system, and for any type of record: blocked or unblocked, fixed or variable length, and undefined.

If GET is used with a file containing checkpoint records, the checkpoint records are bypassed automatically.

FILENAME: GET requires the first operand. The parameter value must be the same as specified in the header entry of the DTF for the file from which the record is to be retrieved. The filename can be specified as a symbol or in either special or ordinary register notation.

WORKNAME: This is an optional parameter specifying the workarea name or a register (in either special or ordinary register notation) containing the address of the workarea. The workarea address should never be preloaded into register 1. This parameter is used if records are to be processed in a workarea that the user himself defines (for example, using a DS instruction). If the operand is specified, all GETs to the named file must always use a register or a workname. Using the second operand causes GET to move each individual record from the input area to a work area.

All records from a logical file may be processed in the same work area, or different records from the same logical file may be processed in different work areas. In the first case, each GET for the file specifies the same work area. In the second case, different GET instructions specify different work areas. It might be advantageous to plan two work areas, for example, and to specify each area in alternate GET instructions. This would permit the programmer to compare each record with the preceding one, for a control change. Only one work area can be specified in any one GET, however.

Required DTF Entries

The input area must be specified in the entry IOAREA1 of the DTF macro. For any file other than a combined file, two input areas may be used to permit an overlap of data transfer and processing operations. The second area is specified in IOAREA2. Whenever two input areas are specified, the IOCS routines transfer records alternately to each area. They completely handle this "flip-flop" so that the next sequential record is always available to the problem program for processing.

For a combined file, the input area is specified in IOAREA1 and the output area in IOAREA2. If the same area is to be used for both input and output, IOAREA2 is omitted.

When records are processed in the input area(s), a register must be specified in the entry IOREG of the DTF macro if:

1. Records are blocked;
2. Variable-length magnetic tape records are read backwards; or

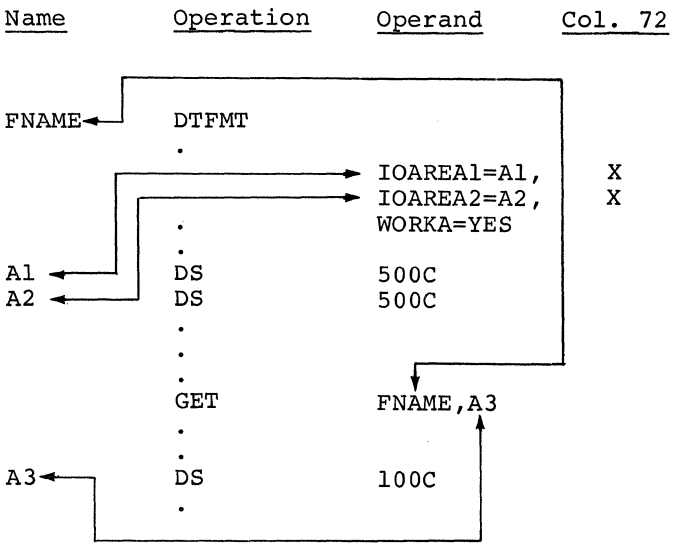
3. Two input areas are used, for either blocked or unblocked records.

This register identifies the next single record to be processed. It always contains the absolute address of the record currently available. The GET routine places the proper address in the register.

If a work area is used, WORKA=YES must be specified. IOREG should not be specified.

When the GET macro detects an end-of-file condition, IOCS branches to the user's end-of-file routine (specified by EOFADDR).

An example of GET/PUT processing is shown in the following coding. The parameter IOAREA1 points to the first I/O area for this file; IOAREA2 points to the second I/O area. GET points to the file-definition block and to the work area (A3) to which logical records are moved from areas A1 and A2 by LIOCS.



Unblocked Records

Records retrieved from any input file are considered fixed unblocked unless otherwise specified.

Whenever records are unblocked (either fixed or variable length) and only one input area is used, each GET transfers a single record from an I/O device to the input area, and then to a work area if one is specified in the GET instruction. If two input areas are specified, each GET makes the last record that was transferred to main storage available for processing in the input area or work area. The same GET also starts the transfer of the following record to the other input area.

When an IBM 2540 Card Read-Punch is used for a card input file, each GET instruction normally reads the record from a card in the read feed. However, if the 2540 has the punch-feed-read special feature installed and if CMBND is specified in the entry TYPEFLE, each GET reads the record from a card in the punch feed, at the punch-feed-read station. This record can be updated by additional information and punched back into the same card, when that card passes the punch station and a PUT instruction is issued. (See PUT: Updating.)

Blocked Records

When records on DASD or magnetic tape are specified as blocked in the entry RECFORM, each individual record must be located for processing (deblocked). Therefore, blocked records (either fixed or variable length) are handled as follows:

1. The first GET instruction transfers a block of records from DASD or tape to the input area. It also initializes the specified register to the absolute address of the first data record, or it transfers the first record to the specified work area.

2. Subsequent GET instructions either add an indexing factor to the register or move the proper record to the specified work area, until all records in the block have been processed.

3. Then the next GET makes a new block of records available in main storage, and either initializes the register or moves the first record.

Undefined Records

When undefined records are to be handled, the entry RECFORM=UNDEF must be included in the file definition. GET treats undefined records as unblocked, and the programmer must locate individual records and fields. If a RECSIZE register is specified, IOCS will store the length of the record read in that register. Undefined records are considered to be variable in length by IOCS. No other characteristics of the record are known by IOCS. They are the responsibility of the user.

Read Backwards, Tape

If records on magnetic tape are to be read backwards (BACK specified in entry READ), blocks of fixed-length records, blocks of blocked-variable records, or unblocked records, are transferred from tape to main storage in reverse order. The last block is

read first; the next-to-last block, second; etc. For blocked records, each GET instruction also makes the individual records available in reverse order. The last record in the input area is the first record available for processing (either by indexing or in a work area).

If a tapemark precedes the data records at the beginning of the tape, a nine-track tape can be read backwards. Seven-track tape can be read backwards only if:

- the tape was originally written on a magnetic tape unit of the IBM System/360;
- the Data Conversion special feature was not used when the tape was written; and
- a tape mark was written at the beginning of the tape preceding the data records.

PUT MACRO

Name	Operation	Operand
[name]	PUT	{Filename / (1)} [, {Workname / (0)}]

PUT writes or punches logical records that have been built directly in the output area or in a specified work area. It is used for any output file in the system, and for any type of record: blocked or unblocked, fixed or variable length, and undefined. It operates much the same as GET but in reverse. It is issued after a record has been built.

FILENAME: PUT requires the first operand. The parameter value must be the same as specified in the header entry of the DTF for the file being built. The filename can be specified as a symbol or in either special or ordinary register notation.

WORKNAME: An optional parameter specifying the work area name or a register (in either special or ordinary register notation) containing the address of the work area. The work area address should never be preloaded into register 1. This parameter is used if records are to be built in a work area that the user himself defines (for example, using a DS instruction). If the operand is specified, all PUTs to the named file must always use a register or a workname. Using the second operand causes PUT to move each record from the work area to the output area.

Individual records for a logical file may be built in the same work area or in different work areas. Each PUT instruction specifies the work area where the completed record was built. However, only one work area can be specified in any one PUT instruction.

Whenever an output data record is transferred from an output area (or work area) to an I/O device (by a PUT instruction), the data remains in the area until it is either cleared or replaced by other data. IOCS does not clear the area. Therefore, if the user plans to build another record whose data does not use every position of the output area or work area, he must clear that area before he builds the record. If this is not done, the new record will contain interspersed characters from the preceding record. For example, in the case of output to a printer, the forms design may require printing in selected positions on one print line and in different positions on another line. In this case, the output area or work area for the printer file should be cleared between lines.

Required DTF Entries

The output area must be specified in the entry IOAREA1 of the DTF macro. For any file other than a combined file, two output areas may be used to permit an overlap of data transfer and processing operations. The second area is specified in IOAREA2. Whenever two output areas are specified, th IOCS routines transfer records alternately from each area. The routines completely handle this "flip-flop", so that the proper output record area is always available to the program for the next sequential output record.

For a combined file, the input area is specified in IOAREA1 and the output area in IOAREA2. If the same area is to be used for both input and output, IOAREA2 is omitted.

When records are built in the output area(s), a register must be specified in th entry IOREG if:

1. Records are blocked, or
2. Two output areas are used, for either blocked or unblocked records.

This register always contains the absolute base address of the currently available output-record area. IOCS places the proper address in the register.

The user should always address the I/O areas by using the IOREG as the base register and should not make any assumption as to which I/O area is presently being used.

If a work area is used, WORKA=YES must be specified; IOREG should not be specified

If the blocked records are variable length and are being built in the output area(s), an additional register must be

specified in the entry VARBLD. IOCS stores the number of bytes remaining in the output area in the VARBLD register each time a PUT instruction is executed.

Unblocked Records

Records transferred to any output file except DASD or magnetic tape are always considered fixed unblocked unless otherwise specified. Records for DASD or tape output are treated as unblocked if this is specified in the entry RECFORM.

Whenever records are unblocked (either fixed or variable length), each PUT transfers a single record from the output area (or input area if updating is specified) to the file. If a work area is specified in the PUT instruction, the record is first moved from the work area to the output area (or input area) and then to the file.

For fixed DASD unblocked records, IOCS uses the rule that if there is not enough space for another record in the extent specified, then there is not enough space for an EOF record.

Blocked Records

When blocked records are to be written on DASD or magnetic tape, the individually built records must be formed into a block in the output area. Then the block of records is transferred to the output file. The blocked records may be either fixed or variable length.

Fixed-length blocked records can be built directly in the output area or in a work area. Each PUT instruction for these records either adds an indexing factor to the register IOREG, or moves the completed record from the specified work area to the proper location in the output area. When an output block of records is complete, PUT causes the block to be transferred to the output file and initializes the register if one is used.

Variable-length blocked records can also be built in either the output area or a work area. The length of each variable-length record must be determined by the problem program and included in the output record as it is built. The problem program can calculate the length of the output record from the length of the corresponding input records. That is, variable-length output records are generally developed from previously written variable-length input records, perhaps modified by current records. Each variable-length input record must include the field that contains the length of the record.

When variable-length blocked records are built in a work area, the PUT instruction performs approximately the same functions as it does for fixed-length blocked records. The PUT routines check the length of each output record to determine if the record will fit in the remaining portion of the output area. If the record will fit, PUT immediately moves the record. If it will not fit, PUT causes the completed block to be written and then moves the record. Thus, this record becomes the first record in a new block.

If variable-length blocked records are to be built directly in the output area, however, the VARBLD entry, the TRUNC macro, and additional user programming are required. The user's program must determine if each record to be built will fit in the remaining portion of the output area. This must be known before processing of the record is started, so that if the record will not fit, the completed block can be written and the record can be built at the beginning of a new block. Thus, the length of the record must be pre-calculated and compared with the amount of remaining space.

The amount of space available in the output area at any time can be supplied to the program (in a register) by the IOCS routines. For this, the user must specify a general-purpose register in the DTF entry VARBLD. This register is in addition to the register specified in IOREG. Each time a PUT instruction is executed, IOCS loads into this register the number of bytes remaining in the output area. The problem program uses this to determine if the next variable-length record will fit. If it will not fit, a TRUNC macro instruction must be issued to transfer the block of records to the output file and make the entire output area available for building the next block.

Undefined Records

When undefined records are handled, PUT treats them as unblocked. The programmer must provide any blocking he wants. He must also determine the length of each record (in bytes) and load it in a register for IOCS use, before he issues the PUT instruction for that record. The register that will be used for this purpose must be specified in the DTF entry RECSIZE.

Updating

A sequential file on 2311 or 2321 DASD, a card input file in a 1442 or 2520, or a card file in the punch feed of a 2540 equipped with the punch-feed-read special feature can be updated. That is, each DASD

or card, from which it was read. In the case of a card file, the file must be specified as a combined file (CMBND) in the entry TYPEFLE.

When updating a file, one I/O area can be specified (entry IOAREA1) for both the input and output of a card record. If a second I/O area is required, it can be specified with IOAREA2.

A PUT for a card or DASD record must always be followed by a GET before another PUT is issued. GETs can be issued as many times in succession as desired.

For a file in a 2540 with the punch-feed-read special feature, a PUT instruction must be issued for each card. A PUT instruction may be omitted, however, if a particular card does not require punching in a 1442 or 2520.

In the following example, data will be punched in the same card that was read. Information from each card is read, processed, and then punched into the card to produce an updated record.

Name	Operation	Operand	Col 72
FILEC	DTFCD		X
		TYPEFLE=CMBND,	X
		IOAREA1=AREA,	X
		DEVADDR=SYS005,	X
		RECFORM=FIXUNB,	X
		IOAREA2=AREA2	
	o		
	o		
	o		
	GET	FILEC	
	o		
	o		
	o		
	PUT	FILEC	
	o		
	o		

RELSE MACRO

Name	Operation	Operand
[name]	RELSE	{Filename / (1)}

The RELSE (release) macro instruction is used in conjunction with blocked input records read from DASD or magnetic tape. It allows the programmer to skip the remaining records in a block and continue processing with the first record of the next block when the next GET instruction is issued.

The symbolic name of the file, specified in the DTF header entry, is the only parameter required for this instruction. It can be specified as a symbol or in register notation.

The release instruction discontinues the deblocking of the present block of records, which may be either fixed or variable length RELSE causes the next GET instruction to transfer a new block to the input area, or switch I/O areas, and make the first record of the next block available for processing. GET initializes the register or moves the first record to a work area.

For example, this function can apply to a job in which records on DASD or tape are categorized, and each category (perhaps a major grouping) is planned to start as the first record in a block. For selective reports, specified categories can be located readily by checking only the first record in each block.

TRUNC MACRO

Name	Operation	Operand
[name]	TRUNC	{Filename / (1)}

The TRUNC (truncate) macro instruction is used in conjunction with blocked output records that will be written on DASD or magnetic tape. It allows the programmer to write a short block of records. (Blocks do not include padding.) Thus the TRUNC macro can be used for a function similar to the RELSE instruction for input records, but in reverse. That is, when the end of a category of records is reached, that block can be written and the new category can be started at the beginning of a new block.

The symbolic name of the file, specified in the DTF header entry, is the only parameter required in this instruction. If this macro will be issued for fixed-length blocked DASD records, the DTF entry TRUNC must be included in the file definition.

When TRUNC is issued, the short block is written (on DASD or tape) and the output area is made available to build the next block. The last record included in the short block is the record that was built before the last PUT instruction preceding TRUNC was executed. Therefore, if records are built in a work area and the problem program determines that a record belongs in a new block, the TRUNC instruction should be issued first, followed by the PUT instruction for this particular record. If records are built in the output area, however, the programmer must determine if a record belongs in the block before he builds the record.

Whenever variable-length blocked records are built directly in the output area, this TRUNC instruction must be used to write a complete block of records. When the PUT instruction is issued after each variable-length record is built, the output routines supply the programmer with the space (number of bytes) remaining in the output area. From this the programmer determines if his next variable-length record will fit in the block. If it will not fit, he issues the TRUNC instruction to write out the block and make the entire output area available to build the record. The amount of remaining space is supplied in the register specified in the entry VARBLD (see PUT Macro and DTFMT VARBLD).

PUNCH AND PRINTER CONTROL

Stacker selection in a card read-punch, and line spacing or skipping in a printer, can be controlled either by specified control characters in the data records or by the CNTRL macro instruction. Either method, but not both, may be used for a particular logical file.

When control characters in data records are to be used, the DTF entry CTLCHR must be specified, and every record must contain a control character in the main-storage output area. This must be the first character of each fixed-length or undefined record, or the first character following the record-length field in a variable-length record. The BLKSIZE specification for the output area must include the byte for the control character and, if undefined records are specified, the RECSIZE specification must also include this byte.

When a PUT instruction is executed, the control character in the data record determines the command code (byte) of the Channel Command Word (CCW) that IOCS establishes.

If CTLCHR=ASA: the control character is translated into the command code.

If CTLCHR=YES: the control character is used directly as the command code.

The particular character included in the record is determined by the function to be performed. For example, if double spacing is to occur after a particular line is printed, the code for double spacing must be the control character in the output line to be printed. The first character after the control character in the output data becomes the first character punched or printed. A complete listing of the control characters is given in Appendix B.

CNTRL MACRO

Name	Operation	Operand
[name]	CNTRL	{Filename / (1)},code[,n][,m]

The CNTRL (control) macro instruction provides commands for these input/output units: magnetic tape units, card read-punches, punches, printers, DASD, and optical readers. Orders apply to physical nondata operations of a unit and are peculiar to the unit involved. They specify such functions as rewinding tape, card stacker selection, line spacing on a printer, etc. For optical readers orders, specify marking error lines or keyboard correcting a line for journal tapes, document stacker selecting, or ejecting and incrementing documents. The CNTRL macro does not wait for completion of the order before returning control to the user, except for certain mnemonics for optical readers.

CNTRL is used in conjunction with a logical file in a unit, and it usually requires two or three parameters. The first parameter must be the name of the file specified in the DTF header entry. It can be specified as a symbol or in register notation.

The second parameter is the mnemonic code for the command to be performed. This must be one of a set of predetermined codes (Figure 6).

The third parameter n is required whenever a number is needed for stacker selection or immediate printer carriage control. The parameter m applies to delayed spacing or skipping. In the case of a printer file, the parameters n and m may be required.

The CNTRL macro instruction must not be used for printer or punch files if the data records contain control characters and the entry CTLCHR is included in the file definition.

Whenever CNTRL will be issued in the problem program, the DTF entry CONTROL must be included (except for DTFMT) and CTLCHR must be omitted. If control characters are used when CONTROL is specified, the control characters are ignored and treated as data.

MAGNETIC TAPE UNIT CODES

The CNTRL macro instruction is used to control magnetic-tape functions that are not concerned with reading or writing data on

Unit	Mnemonic Code	n	m	Command
2400 Series Magnetic Tape Units	REW			Rewind Tape
	RUN			Rewind and Unload Tape
	ERG			Erase Gap (Writes Blank Tape)
	WTM			Write Tape Mark
	BSR			Backspace to Interrecord Gap
	BSF			Backspace to Tape Mark
	FSR			Forward Space to Interrecord Gap
	FSF			Forward Space to Tape Mark
2540 Card Read	PS	1 2 3		Select Pocket 1, 2, or 3
2520, 1442 Card Read Punch	SS	1 2		Select Stacker 1 or 2
	E			Eject to Stacker 1 (1442 Only)
		See Note		
1403, 1404, 1443, 1445 Printers	SP	c	d	Carriage Space 1, 2, or 3 Lines
	SK	c	d	Skip to Channel c and/or d
1403 Printer with Universal Character Set Feature	UCS	ON		Disallow Data Checks
		OFF		Allow Data Checks
2321 Data Cell Drive	SEEK			Seek to Address
	RESTR			Return Strip to Sub-Cell
2311 Disk Storage Drive	SEEK			Seek to Address
1285 Optical Reader	MARK			Mark Error Line
	READKB			Read 1285 Keyboard
1287 Optical Reader	MARK			Mark Error Line in Tape Mode
	READKB			Read 1287 Keyboard in Tape Mode
	EJD			Eject Document
	SSD	1 2 3 4		Select Stacker A, B, Reject, or Alternate Stacking Mode
	ESD	1-4		Eject Document and Select Stacker
	INC			Increment Document at Read Station.

Note: c = An Integer Indicating Immediate Printer Control (Before Printing).
d = An Integer Indicating a Delayed Printer Control.

Figure 6. CNTRL Macro Instructions

the tape. These functions are grouped in the following categories:

Rewinding tape to the load point
- REW - Rewind
- RUN - Rewind and unload

Moving tape to a specified position
- BSR - Backspace to interrecord gap
- BSF - Backsapce to tape mark
- FSR - Forward space to interrecord gap
- FSF - Forward space to tape mark

Writing a tape mark
- WTM - Write tape mark

Erasing a portion of the tape
- ERG - Erase gap (writes blank tape)

The tape rewind (REW and RUN) and tape movement (BSR, BSF, FSR, and FSF) functions can be used before a tape file is opened. This allows the tape to be positioned at a desired location for opening a file under conditions such as:

- The file is located in the middle of a multifile reel.
- The entry REWIND specifies NORWD, but for some conditions rewinding is required for the file.

The tape movement functions (BSR, BSF, FSR, and FSF) apply to input files only, and the following factors should be considered:

1. The FSR (or BSR) function permits the user to skip over a physical tape record (from one interrecord gap to the next). The record is passed without being read into main storage. The FSF (or BSF) function permits the user to skip to the end of the logical file (identified by a tape mark).

2. The functions of FSR, FSF, BSR, and BSF always start at an interrecord gap.

3. If blocked input records are being processed and if the user does not want to process the remaining logical records in the block, as well as one or more succeeding blocks (physical records), he must issue a RELSE macro before the CNTRL macro. Then the next GET will make the first record of the new block available for processing. If the CNTRL macro, with FSR for example, were issued without a preceding RELSE, the tape would be advanced, but the next GET would make the next record in the old block available for processing.

4. For any I/O area combination except one I/O area and no work area, IOCS is always reading one physical tape record ahead of the one that is being processed. Thus, the next physical record (block) after the one being processed will be in main storage ready for processing. Therefore, if a CNTRL FSR function is performed, the second physical tape record beyond the present one will be passed without being read into main storage.

5. If FSR or BSR is used, LIOCS does not update the block count. Furthermore, IOCS cannot sense tapemarks on an FSR or BSR command and, therefore, does not perform the usual EOV or EOF functions.

PRINTER CODES

The CNTRL macro instruction can be used for any printer forms control.

The CNTRL macro codes for printer operation cause spacing (SP) over a specified number of lines or skipping (SK) to a specified location on the form (represented by a carriage-tape channel). The third parameter is required for immediate spacing and skipping (before printing). The fourth parameter is required for delayed spacing or skipping (after printing).

The SP and SK operations can be used in any sequence. However, two or more consecutive immediate skips (SK) to the same carriage channel on the same printer have the same effect as the first skip only. That is, any skip order after the first is ignored. Two or more consecutive delayed spaces (SP) and/or skips (SK) to the same printer result in the last space or skip only. Any other combination of consecutive controls (SP and SK), such as immediate space followed by a delayed skip or immediate space followed by another immediate space, causes both specified operations to occur.

1403 PRINTER WITH UNIVERSAL CHARACTER SET CODES

The CNTRL macro can be used before a PUT for the file to change the method of processing data checks. They can be either:

1. allowed--indication given to operator, or
2. disallowed--ignored and blank printed.

A data check occurs on a 1403 with the UCS feature when a character (except null, 00000000, or blank, 01000000) sent to the printer does not match any of the characters in the UCS buffer.

If OPEN is used for the file, data checks are automatically disallowed until a CNTRL macro is used to allow them.

If the UCS form of the CNTRL macro is used for a printer without the UCS feature, the CNTRL macro is ignored and the output operation is performed.

2540 CARD READ PUNCH CODE

Cards read or punched on the 2540 normally fall into the pocket specified in the DTF entry SSELECT (or the R1 or P1 pocket if SSELECT is omitted). The CNTRL macro with code PS is used to select a card into a different stacker, which is specified by the third operand (n) of the CNTRL macro. The possible selections are:

Feed	Pocket	Value of n
Read	R1	1
Read	R2	2
Read	RP3	3
Punch	P1	1
Punch	P2	2
Punch	RP3	3

INPUT FILE: CNTRL can be used only when one I/O area, with or without a work area, is specified for the file. The macro is issued after GET. Once used, CNTRL must be used with every succeeding GET for the file.

OUTPUT OR COMBINED FILE: CNTRL can be used with any permissible combination of I/O and work areas. When the user wants to select a particular card, CNTRL must be issued before the PUT for that card. However, CNTRL does not have to precede every PUT.

2520 CARD READ PUNCH CODE

The CNTRL macro used for the IBM 2520 uses the code SS. Thus, the third operand can be either 1 or 2.

1442 CARD READ PUNCH CODES

Cards fed in the IBM 1442 are normally directed to the stacker specified in the DTF entry SSELECT. If SSELECT is omitted, they go to stacker 1. The CNTRL macro can be used to override the normally selected pocket temporarily.

CARD READING: To stack a particular card, the CNTRL macro instruction should be issued after the GET for that card, and before the GET instruction for the following card. When the next card is read, the first card is stacked in the specified stacker. CNTRL can be used only when one I/O area, or one I/O area and one work area is specified for the file.

CARD PUNCHING: To stack a particular card, the CNTRL macro should be issued before the PUT for that card. After the card is punched, it is stacked into the specified pocket immediately. CNTRL can be used with any permissible combination of I/O and work area.

COMBINED FILE: If a particular card is to be selected, the CNTRL macro for the file should be issued after the GET and before the PUT for the card. When the next card is read, the first card is stacked into the specified stacker.

2311 DISK STORAGE DRIVE CODE

The CNTRL macro for seeking on the 2311 allows the user to specify a track address to which access movement should begin for the next GET or PUT instruction for a file. While the arm is moving, the programmer may process data and/or request I/O operations on other devices.

2321 DATA CELL DRIVE CODES

The CNTRL macro enables the user to seek to a specific address or to restore the strip to its cell. The seek address must be provided in the field with the symbolic name given in the DTFDA entry SEEKADR.

IBM 1285 OPTICAL READER AND 1287 OPTICAL READER CODES

Use of the CNTRL macro instruction with the READKB mnemonic allows the user to read a complete line from the 1285 or 1287 keyboard when processing journal tapes. This permits the operator to key in a complete line on the keyboard if a 1285 or 1287 read error makes this type of correction necessary. If the operator cannot readily identify the unreadable character, he may enter a reject character (@) in the error line. IOCS then exits to the user's COREXIT routine, in which he may issue the CNTRL macro instruction to read from the keyboard. The 1285 or 1287 display tube then displays the full line and the operator keys in the correct line from the keyboard, if possible. The line read from the keyboard is always read into the correct area.

The CNTRL macro with the READKB mnemonic waits for completion of the order before returning control to the user.

When processing journal tapes, use of the CNTRL macro instruction with the MARK mnemonic provides a program-controlled

means of marking a line on the input tape that results in a data transfer error or is otherwise suspect of error. To ensure that the proper line is marked, the CNTRL macro instruction must be issued in the user's error correction routine (specified in DTFOR COREXIT). If CNTRL is issued at another time, the line following the one in error will be marked.

When processing in document mode on the 1287, each document must be explicitly ejected with a CNTRL macro instruction. The EJD mnemonic causes the document to be ejected and the next document to be fed.

Documents must also be explicitly stacker selected using the CNTRL macro instruction with the SSD mnemonic. A document may be directed to stacker A, B, or R (reject stacker) by specifying a selection number of 1, 2, or 3 respectively. Also, documents may be selected into stackers A and B in an alternate stacking mode, with automatic stacker switching when one stacker becomes full. The selection number for alternate mode is 4. If selection number 4 is used in the first stacker selection macro, stacker A will be filled first. If selection number 4 is used after other selection numbers, the last preceding selection number determines the first stacker to be filled.

Ejection and stacker selection of documents must occur alternately.

The CNTRL macro instruction with the ESD mnemonic combines the ejection and stacker selection functions. In order to satisfy the alternate ejection and stacker selection functions, the combined mnemonic must not be immediately preceded by an eject or immediately followed by a stacker select.

The CNTRL macro with the INC mnemonic is used for document incrementation. This macro is not used with documents having a scannable area shorter than 6 inches. The document is incremented forward 3 inches. It may be used only once per document.

DSPLY MACRO

Name	Operation	Operand
[name]	DSPLY	Filename,r,r

The DSPLY macro displays the document field on the 1287 display scope. A complete field may be keyboard-entered if a 1287 read error makes this type of correction necessary. An unreadable character may be replaced by a reject character (@) by either the operator (if processing in the on-line correction mode) or by the device (if processing in the off-line correction mode). The user may then use the DSPLY macro to display the field in error. The 1287 display tube displays the full field and the operator keys in the correct field from the keyboard, if possible. The field read from the keyboard is always read into the portion of IOAREA1 originally specified for the field. The macro blanks this portion of IOAREA1. When the operation is completed, the field is left justified in this area.

This instruction always requires three parameters. The first parameter is the symbolic name specified in the DTFOR header entry for the 1287 file. The second parameter specifies a general purpose register (2-12) into which the problem program will have placed the address of the Load Format CCW giving the document co-ordinates for the field to be displayed. The third parameter specifies a general purpose register (2-12) into which the problem program will have placed the address of the Load Format CCW giving the co-ordinates of the reference mark associated with the field to be displayed.

Note: When using the DSPLY macro, the user must ensure that the Load Format CCW giving the document co-ordinates for the field to be displayed (second parameter) is command chained to the Read Backward (or Read Backward and Test) CCW for that field.

READ MACRO

The READ macro instruction is used in sequential processing to cause the next sequential 1287 Optical Reader (document mode only) record to be read.

Name	Operation	Operand
[name]	READ	Filename,SQ,name (r)

The first parameter specifies the name in the DTFOR header for this file, and it is always required.

The parameter SQ is always required.

The parameter name is always required. This parameter specifies the address of the user-provided channel command word list to be used to read a document from the 1287 file. The register entry may be used in this parameter to provide the address of the channel command word list.

RESCN MACRO

Name	Operation	Operand
[name]	RESCN	Filename,r,r

The RESCN macro selectively re-reads a field on a document when a defective character makes this type of operation necessary. The field read is always read into the portion of IOAREA1 originally intended for the field.

This instruction always requires three parameters. The first parameter specifies the symbolic name of the 1287 file specified in the DTFOR header entry for this file. The second parameter specifies a general purpose register (2-12) into which the problem program will have placed the address of the Load Format CCW giving the document co-ordinates for the field to be read. The third parameter specifies a general purpose register (2-12) into which the problem program will have placed the address of the Load Format CCW giving the co-ordinates of the reference mark associated with the field to be read.

Note: When using the RESCN macro, the user must ensure that the Load Format CCW giving the document co-ordinates for the field to be read (second parameter) is command chained to the Read Backward (or Read Backward and Test) CCW for that field.

The user determines whether the read operation generated by RESCN has resulted in a more satisfactory read than the original read of the field. If the re-read of the field results in a wrong length record or a lost line condition, an indication of the reason is provided in Filename+54. See description of COREXIT for hexadecimal values.

RDLNE MACRO

Name	Operation	Operand
[name]	RDLNE	Filename

The RDLNE macro provides selective on-line correction when processing journal tapes on the IBM 1285 or the IBM 1287 Optical Reader. This macro reads a line in the on-line correction mode while processing in the off-line correction mode. If the reader cannot read a character, IOCS retries the line containing the unread character. If still unsuccessful, the user is informed of the condition via his error correction routine (specified in DTFOR COREXIT). The RDLNE macro may then be issued to cause another attempt to read the line. If the character in the line cannot be read during this attempt, the character is displayed on the 1285 or 1287 display scope. The operator may key in the correct character, if possible. If the operator cannot readily identify the defective character, he may enter a reject character (@) in the error line. This condition is posted in "filename+54" for user examination. Wrong length records and lost line conditions are also posted to "filename+54". See the description of COREXIT for hexadecimal values. RDLNE should be used in COREXIT only or the line following the one in error will be read in on-line correction mode.

The macro requires only one parameter, the symbolic name of the 1285 or 1287 file from which the record is to be retrieved. This name is the same as that specified in the DTFOR header entry for this file.

WAITF MACRO

Name	Operation	Operand
[name]	WAITF	Filename

The WAITF macro instruction is used in sequential processing to ensure that the transfer of a 1287 Optical Reader record (document mode only) has been completed. It requires only one parameter: the symbolic name of the file containing the record.

This instruction must be issued before the problem program attempts to process an input record for the file concerned. The program enters a waiting loop until the transfer of data is complete. Thus, the WAITF macro instruction must be issued after any READ instruction for a file, and before the succeeding READ instruction for the same file.

The WAITF macro instruction accomplishes all checking for read errors on the 1287 file and exits to the user-provided COREXIT routine for user handling of these conditions, if necessary.

CHNG MACRO

Name	Operation	Operand
[name]	CHNG	SYSnnn

This macro instruction is provided only for Basic Programming Support and Basic Operating System upward compatibility. No code is generated from this macro instruction. In the Disk Operating System, tape

channel switching is handled automatically by Physical IOCS.

PRTOV MACRO

Name	Operation	Operand
[name]	PRTOV	{Filename / (1)} , {9 / 12} [, {Routine-name / (0)}]

The PRTOV (printer overflow) macro instruction is used in conjunction with a logical file in a printer to specify the operation to be performed on a carriage overflow condition. Whenever this macro instruction is to be issued in a problem program, the DTFPR or the DTFSR entry PRINTOV=YES must be included in the file definition.

PRTOV requires two or three parameters. The first parameter must be the filename as a symbol or in register notation. The second parameter must specify the number of the carriage tape channel (9 or 12) used to indicate the overflow. When overflow condition occurs, IOCS restores the printer carriage to the first printing line on the form (channel 1), and printing of detail lines continues.

A third parameter is required if the programmer prefers to branch to his own routine on an overflow condition, rather than skipping directly to channel 1 and continuing with the detail printing. It specifies the symbolic name of the user's routine. The name can be specified as a symbol or in register notation. However, the name should never be preloaded into register 1.

In this case, IOCS does not restore the carriage to channel 1. In his routine, the user may issue any IOCS macro instruction (except another PRTOV) to perform whatever functions he desires. The CNTRL macro cannot be issued to the file unless CONTROL=YES has been specified in the DTF. For example, he can print total lines, skip to channel 1, and print overflow page headings. At the end of his routine, the user must return to IOCS by branching to the address in register 14. IOCS supplies this address upon entry to the user's routine. Therefore, if other IOCS macros are used in the routine (for example, the CNTRL macro), the user must save and restore register 14 himself.

The PRTOV macro causes a skip to channel 1, or branches to the user's routine if an overflow condition (punch in channel 9 or 12) is detected on the preceding space or print command. An overflow punch is not recognized during a carriage skip operation. After the execution of any command that causes carriage movement (PUT or immediate CNTRL), the user should issue a PRTOV macro before issuing the next CNTRL or PUT. This ensures that the user's overflow option will be executed at the correct time.

MACROS FOR TAPE AND DISK WORK FILES

A work file can be used for input, output, or both. If TYPEFLE=WORK is specified in the DTF macro instruction, work-file macro instructions READ, WRITE, and CHECK are provided. In addition, if NOTEPNT is specified, workfile macro instructions NOTE, POINTR, POINTW, and POINTS are provided. Work files will handle fixed-length unblocked records and undefined-format records.

WORK FILES ON TAPE

A work file is a single-volume file that can be used for both input and output, even within a single program phase. It is often used to pass intermediate results between successive phases or job steps. However, work files also can be written, read, and rewritten within a single phase, without requiring additional OPEN or CLOSE processing. Work files are defined as an option of the DTFMT and MTMOD macro instructions and are accessed by the READ/WRITE/CHECK macro instructions.

The first time a work file is opened, it is opened as an output file. OPEN examines the tape to determine whether the tape used contains standard labels. The DTFMT entry FILABL is ignored. If the tape is labeled and the date in the header label is expired, a new label consisting of HDR1 and followed by 76 binary zeros is created. The Job Control label information cards are not required. If the tape does not already contain standard labels, labels are not created for the work file. Trailer labels are not processed.

If a work file with standard labels is reopened, OPEN determines from the HDR label that the file is a work file and does not rewrite the labels.

When a tapemark is sensed during a read operation or when an end-of-reel reflective spot is sensed during a write operation, the IOCS work file logic exits to the address specified by the user in the entry EOFADDR.

WORK FILES ON DISK

Disk (2311) work files are supported as single-volume single-pack files. They are always opened as output files. Standard label information must be supplied by the

user. Both normal extents (type 1) and split extents (type 128) are supported. File protection for work files is ensured only if their labels are unexpired.

WORK FILE TO BE DELETED AFTER USE: The entry DELETFL=NO must not be used. OPEN creates a Format 1 label for the file, and CLOSE destroys the label. The next job requiring a work file can use the same extents and filename.

WORK FILE TO BE SAVED AFTER USE: The expiration date in the DLAB Job Control card must not be the current date. The entry DELETFL=NO must be specified in the DTF for the file. OPEN creates a Format 1 label, but CLOSE does not delete it. Thus the file can be saved until the expiration date is reached.

DELETING AN UNEXPIRED FILE: When the user tries to use the limits of an unexpired file, a message is printed to indicate the overlap, and the operator can delete the label. Then OPEN will create a label for the new file and the job will continue.

The following text discusses the macro instructions used with a magnetic tape or disk (2311) work file.

READ MACRO

The READ macro instruction will cause the next sequential physical record, or part of it, to be read from the file associated with the Filename, into the area of main storage indicated by the third operand.

The DTF entry READ=FORWARD or BACK should be used to specify the type of read for a tape file.

Name	Operation	Operand
[name]	READ	{Filename / (1)}, SQ, {area / (0)} [, {length / (r) / S}]

The first parameter specifies the name of the file associated with the record to be read and is always required. This name is the same as the name specified in the DTFMT or DTFSD header entry for this file. The name can be given as a symbol or in register notation.

The parameter SQ (for sequential) is always required. Area specifies the name, as a symbol or in register notation, of the input area used by this file. If the tape is to be read backwards, area must be the address of the rightmost byte of the input area.

The length parameter is used only for records of undefined format (RECFORM=UNDEF). To read only a portion of a record, an actual length (or a register containing the number) can be stated. Or, an S can be provided to indicate that the entire physical record should be read.

If the work file records are fixed-length unblocked records (RECFORM=FIXUNB), the length parameter is not specified in the READ macro. The number of characters to be read is specified in the BLKSIZE entry. The user can change this number, which is stored in the DTF table, at any time by referencing the halfword FilenameL.

If the fourth parameter is not specified, the third parameter (area) should never be preloaded into register 0. If the fourth parameter is specified, special register notation is not necessary to allow the macro to be used in a self-relocating program.

WRITE MACRO

The WRITE macro instruction causes a record to be written from the indicated area into the file associated with the Filename. The record will be stored sequentially following the last record written in this file.

Name	Operation	Operand
[name]	WRITE	{Filename / (1)}, {SQ / UPDATE}, {area / (0)} [, {length / (r)}]

The first parameter specifies the name of the file associated with the record to be written and is always required. This name is the same as the name specified in the DTFMT or DTFSD header entry for this file. The name can be given as a symbol or in register notation.

The second parameter specifies the type of WRITE to be executed. For magnetic tape it is always SQ. For 2311 work files if SQ is specified, a formatting WRITE (Write Count Key and Data) will be executed; if UPDATE is specified, a non-formatting WRITE (Write Data) will be executed. An update WRITE will generally follow a READ macro instruction.

The parameter area specifies the name, as a symbol or in register notation, of the output area used by this file.

The last parameter is used only for records of undefined format (RECFORM=UNDEF). Length specifies the actual number (or register containing the number) of bytes to be written.

If fixed length unblocked records (RECFORM=FIXUNB) are being written, length is not used in the write macro. The number of characters to be written is specified in the BLKSIZE entry. The user can change this number, which is stored in the DTF table, at any time by referencing the halfword FilenameL.

If the fourth parameter is not specified, the third parameter (area) should not be preloaded into register 0. If the fourth parameter is specified, special register notation is not necessary to allow the macro to be used in a self-relocating program.

CHECK MACRO

Name	Operation	Operand
[name]	CHECK	{Filename / (1)}

This macro instruction prevents the user from processing until completion of the input/output operation, started by a READ or a WRITE, for the device associated with the Filename. If the I/O operation is completed without any error or other exceptional condition, CHECK returns control to the next instruction. If the operation results in a read error, CHECK will process the user's option specified in ERROPT. If CHECK finds an end-of-file condition, control is passed to the routine specified in EOFADDR.

NOTE MACRO

Name	Operation	Operand
[name]	NOTE	{Filename / (1)}

The NOTE macro instruction is used to obtain identification for the last physical record that was read or written.

For a tape, the identification is the number of physical records that have been read or written in the specified file from the load point. The information is returned in register 1 in the form 0BBB where 0 = eight binary zeros, and BBB = the physical record number in binary.

For a 2311, the identification is returned in register 1 in the form 0CHR, where 0 = eight binary zeros, C = cylinder number, H = track number, R = record number within the track; C, H, and R are binary numbers.

The user may store the identification (in the 0BBB or 0CHR form), and later present it to a POINTR or POINTW macro to find the desired record.

If NOTE follows a WRITE to a 2311 file, the unused space remaining on the track following the end of the identified record is returned in register 0 as the binary number 00LL.

The user must ensure the last operation was completed satisfactorily by using CHECK prior to issuing a NOTE.

POINTR MACRO

The POINTR macro instruction is used to reposition the file to read a record previously identified by a NOTE macro instruction.

Name	Operation	Operand
[name]	POINTR	{Filename / (1)} , {address / (0)}

Address is the address (as a symbol or in register notation) of the 4-byte main-storage location containing the required record identification. Address should never be preloaded into register 1. The 4-byte number must be in the form obtained from the NOTE macro. On magnetic tape, POINTR, followed by a WRITE, causes the identified record to be overwritten (destroyed).

POINTW MACRO

The POINTW macro instruction is used to reposition the file to write a record after one previously identified by a NOTE macro instruction.

Name	Operation	Operand
[name]	POINTW	{Filename / (1)} , {address / (0)}

Address is the address (as a symbol or in register notation) of the main-storage location where the user has the required record identification stored. Address should never be preloaded into register 1.

For tapes, the identification must be in the form obtained from the NOTE macro. When a READ is issued to a tape file following a POINTW, the tape is positioned to read the record following the one identified by the POINTW.

For 2311 work files, the user must build the identification from 0CHR and 00LL into a 6-byte number 0CHRLL. POINTW may be followed by a WRITE only if the space remaining on the track is available. The unused space remaining on the current track will not be obtainable if a NOTE is given after a READ macro instruction.

When using disk work files, the user should consider the following two conditions. First, a READ issued to a disk file following a POINTW reads the same record identified by the POINTW. Second, some programs using disk work files may include multiple WRITE instructions following a NOTE macro. If a POINTW instruction is issued and the work file records are in undefined format, there may be occasions when the replacement record (being longer than the original record) cannot be written in the space available on the track. In this case, when the next WRITE is performed, the original record will remain as the last record on the track, and the replacement record will be written as the first record on the track.

POINTS MACRO

The POINTS macro instruction is used to reposition a file to the beginning of the file.

Name	Operation	Operand
[name]	POINTS	{Filename / (1)}

For a tape file, the tape will be rewound. If any header labels are present, they will be bypassed, and the tape will be positioned to the first record following the label set.

For a 2311, the file will be repositioned to the lower limit of the first extent.

PROCESSING DASD RECORDS BY THE DIRECT ACCESS METHOD

DASD records can be processed in random order by the Direct Access Method (DAM). In this method the user specifies the address of the record to IOCS and issues a READ or WRITE macro instruction to transfer the specified record. Variations in the parameters of the READ or WRITE instructions permit records to be read, written, updated, or replaced in a file. Whenever this method of processing records is used, the logical file and main-storage area(s) allotted to the file must be defined by the declarative macro DTFDA (Define the File for Direct Access).

RECORD TYPES

DASD records that will be processed by DAM can exist on the DASD in either of two formats: with a key area, or without.

With key area:

Without key area:

Whenever records in a file have keys that are to be processed:

- Every record must have a key, and
- All keys must be the same length.

Whenever the DTFDA entry KEYLEN is not specified for a file, IOCS ignores keys, and the DASD records may or may not contain key areas. A WRITE ID or READ ID will read or write the data portion of the record. However, when KEYLEN is not specified in the DTF for the file, WRITE AFTER cannot be used to extend a file which has keys.

IOCS considers all records as unblocked (one logical record per one physical record). If the user wants blocked records, he must provide his own blocking and deblocking. Records are also considered to be either fixed length or undefined. (Variable-length records can be handled by specifying them as undefined.) The type of records in the file must be specified in the DTFDA entry RECFORM. Whenever records specified as undefined are to be written to a file, the user must determine the length of each record and load it in a register (specified by the DTFDA entry RECSIZE) before he issues the WRITE instruction for that record.

DIRECT ACCESS IOAREA1

The DTFDA entry IOAREA1 defines an area of main storage in which records are read on input or built on output.

Format

The format of the I/O area is determined at assembly time by the following DTFDA entries: AFTER, KEYLEN, READID, WRITEID, READKEY, and WRITEKY. Figure 7 describes the types of DTF macros and the I/O areas that they define. The information in this figure should be used to determine the length of the I/O area specified in the BLKSIZE entry. The I/O area must be large enough to contain the largest record in the file. If the DTF used requires it, the I/O area must include room for an 8-byte count field. The count is provided by IOCS.

Contents

The phrase contents of the IOAREA1 refers to the information provided by or to IOCS for a specific imperative macro instruction. See Figure 32 for a summary of what the contents are for each type of READ/WRITE. When the user is building a record, he must place the contents in the appropriate field of the I/O area using Figure 7 as a guide. The contents that IOCS provides on input are always placed in the appropriate field of the I/O area. For example, if the DTF used for the file resulted in the uppermost format shown in Figure 7, the Data would be located to the right of the Count and Key area.

REFERENCE METHODS

With the direct access method of processing, each record that is to be read or written is specified by providing IOCS with two references:

- Track reference. This gives the track on which the desired record is located.
- Record reference. This may be either the record key (if the records contain key areas) or the record identifier (ID).

IOCS seeks the specified track, searches it for the individual record, and reads or writes the record as indicated by the macro instruction. If a specified record is not found, IOCS sets a no-record-found indication in the user's error/status byte, which is specified by the DTFDA entry ERRBYTE. This indication can be tested by the problem program, and additional processing can be programmed to suit the user's requirements.

Multiple tracks can be searched for a record specified by Key (SRCHM). If a record is not found after an entire cylinder or the remainder is searched, an end-of-cylinder bit is turned on instead of NRF in ERRBYTE.

When the I/O operation is started, control is returned immediately to the problem program. Therefore when the program is ready to process the input record, or build the succeeding output record for the same file, a test must be made to ensure that the previous transfer of data is complete. This is done by issuing a WAITF macro instruction in the problem program.

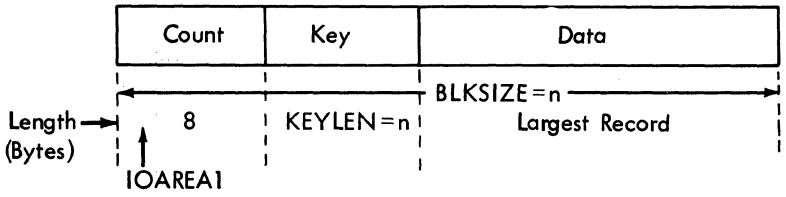
After a READ or WRITE instruction for a specified record has been executed, IOCS can make the ID of the next record available to the problem program after the data transfer has been assured by the use of WAITF. To request that IOCS supply the ID, the user must set up a 5-byte field (in which IOCS can store the ID) and specify the symbolic address of this field in the DTFDA entry IDLOC.

When record reference is by key and multiple tracks are searched, the ID of the specified record (rather than the next record) is supplied. The function of supplying the ID is useful for a random updating operation or for the processing of successive DASD records. If the user is processing consecutively on the basis of the next ID and does not have an end-of-file record, he can check the ID supplied by IOCS against his file limits to determine when he has reached the end of his logical file.

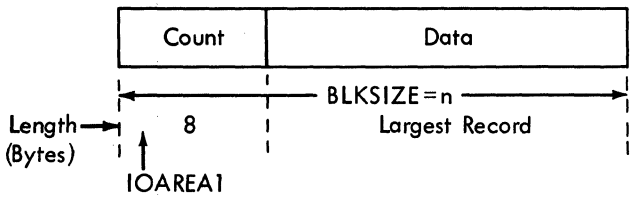
Track Reference

To provide IOCS with the track reference, the user sets up an 8-byte track-reference field in main storage, assigns a symbolic name, and specifies the symbolic name in the DTFDA entry SEEKADR. Before issuing any read or write instruction for a record, the user must store the proper track information (MBBCCHH) in the first seven bytes of this field. The field (Figure 8) contains the following seven bytes for track reference. The eighth byte (R), listed here and shown in the figure, is used when reference to records is by record number (see Record Reference: Identifier). All numbers must be supplied in binary notation.

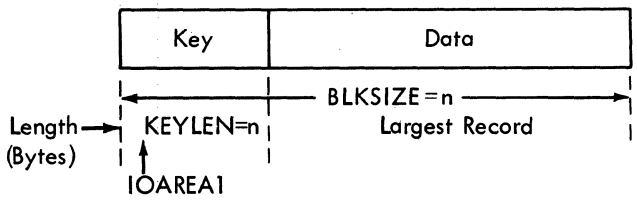
Using a DTF for which AFTER and KEYLEN are specified. READID, WRITEID, READKEY, and WRITEKY may also be specified.



Using a DTF for which AFTER is specified, but KEYLEN is not. READID and/or WRITEID may be specified, but READKEY and WRITEKY may not.



Using a DTF for which KEYLEN, READID, and/or WRITEID are specified, but AFTER is not. READKEY and/or WRITEKY may also be specified.



Using a DTF for which READID and/or WRITEID is specified, but AFTER, KEYLEN, READKEY, and WRITEKY are not.

or

Using a DTF for which KEYLEN, READKEY, and/or WRITEKY are specified, but AFTER, READID, and WRITEID are not.

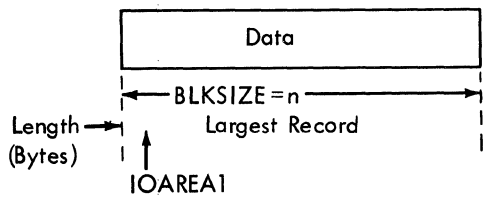


Figure 7. Schematic of I/O Area in Main Storage, for DAM

Byte	Identifier	Contents	Information
0	M	0-244	Number of the pack (0-244) on which the record is located. All packs for a file must be numbered consecutively starting with 0. That is, the first pack must be number 0, the second pack number 1, etc. This number relates to a numbered symbolic unit (SYS000-SYS244). Two or more symbolic units for a file must be numbered consecutively, but the numbering may start with any SYSnnn number.
1-2	B,B	0,0 (for 2311) 0,0-9 (for 2321)	For 2321 the first byte is zero. The cell Number (0-9) is specified in the second byte. These two bytes are always zero for 2311 disk-storage references.
3-4	C,C	0,0-199 (for 2311) 0-19,0-9 (for 2321)	For 2311 the number of the cylinder (0-199) in which the record is located. The first byte is always zero, and the second byte specifies one of the available cylinders in a disk pack. These two bytes with the next two (HH) provide the track identification. For 2321 the number of the subcell (0-19) is located in the first byte. One of the ten strips (0-9) is located in the second byte. Note: The last four strips on each cell are reserved for alternate tracks.
5-6	H,H	0,0-9 (for 2311) 0-4,0-19 (for 2321)	For 2311 the number of the read/write head (0-9) that applies to the record. The first byte is always zero, and the second byte specifies one of the ten disk surfaces in a disk pack. For 2321 the first byte (0-4) specifies one of the five head bar positions (equivalent to cylinder on 2311). The second byte (0-19) specifies one of the twenty head elements.
7	R	0-255	Sequential number of the record on the track. Note: R = 0 if reference is by key.

Figure 8. Track Reference Field

When the READ or WRITE is executed, IOCS refers to this field to select the specific track on the appropriate DASD.

Record Reference

The Direct Access Method allows records to be specified by record key or by record identifier.

Keys

If records contain key areas, the records on a particular track can be randomly searched by their keys. This allows the user to refer to records by the logical control information associated with the records, such as an employee number, a part number, a customer number, etc.

For this type of reference the programmer must specify, in the DTFDA entry KEYARG, the symbolic name of a main-storage key field. He then stores each desired key in this field.

Identifier (ID)

Records on a particular track can be randomly searched by their position on the track, rather than by control information. This is accomplished by using the record identifier (ID). The record identifier, which is part of the count area of the DASD record, consists of five bytes (CCHHR). The first four bytes (cylinder and head) refer to the location of the track and the fifth byte (record) uniquely identifies the particular record on the track. When records are specified by ID, they must be numbered in succession, and without missing numbers, on each track. The first data record on a track must be record number 1, the second number 2, etc.

Whenever records are to be identified by the record ID method, the eighth byte (R) of the track-reference field (Figure 11) must contain the number of the desired record. When a READ or WRITE instruction that searches by ID is executed, IOCS refers to the track-reference field to determine which record is requested by the program. The number in this field is compared with the corresponding fields in the count areas of the disk records. The R-byte specifies the particular record on the track.

CREATING A FILE OR WRITING ADDITIONAL RECORDS ON A FILE

In addition to reading, writing, and updating records randomly, the direct access method permits the user to create a file or write new records on a file. When this is done, all three areas of a DASD record are written: the count area, the key area (if present), and the data area. The new record is written after the last record previously written on a specified track. The remainder of the track is erased. This method is specified in a WRITE instruction by the parameter AFTER.

IOCS ensures that each record will fit on the track specified for it. If the record will fit, IOCS writes the record; if it will not fit, IOCS sets a no-room-found indication in the user's error/status byte (specified by the DTFDA entry ERRBYTE). In the AFTER method IOCS also determines (from the capacity record) the location where the record is to be written.

Whenever the AFTER option is specified, IOCS uses the first record on each track (R0) to maintain updated information about the data records on the track. Record 0 (Figure 9) has a count area and a data area, and contains the following:

Count Area

Flag (not normally transferred to main storage)

Identifier

Key Length (KL)

Data Length (DL)

Data Area (8 bytes)

5 Bytes--ID of last record written on track (CCHHR).

2 Bytes--Number of unused bytes remaining on track.

1 Byte--For the DAM on the Operating System/360.

Each time a WRITE AFTER instruction is executed, IOCS updates the data area of this record.

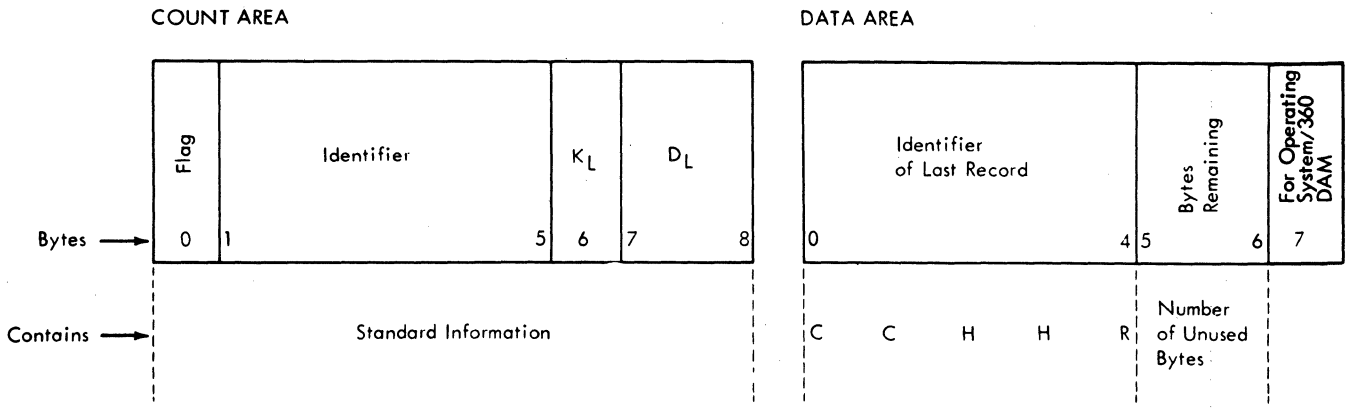


Figure 9. Contents of Record 0 for Capacity-Record Option

READ MACRO

Name	Operation	Operand
[name]	READ	{Filename / (1)} , {KEY / ID}

This instruction causes a record to be transferred from DASD storage to an input area in main storage. The input area must be specified in the DTFDA entry IOAREA1.

The READ macro instruction is written in either of two forms, depending on the type of reference used to search for the record. Both forms may be used for records in any one DTFDA-specified logical file if the logical file has keys.

The instruction always requires two parameters. The first parameter specifies the name of the file from which the record is to be retrieved. This name is the same as that specified in the DTFDA header entry for this file and can be given as a symbol or in register notation. The second parameter specifies the type of reference used for searching the records in the file.

If records in the file are specified as undefined (RECFORM=UNDEF), DAM will supply the data length of each record in the register that is specified in the DTF entry RECSIZE.

Record Reference by Key

If the record reference is by key (control information in the key area of the DASD record), the second parameter in the READ instruction must be the word KEY, and the DTFDA entry READKEY must be included in the file definition.

Whenever this method of reference is used, the problem program must supply the key of the desired record to IOCS before the READ instruction is issued. For this, the key must be stored in the key field (specified in the DTFDA entry KEYARG). When the READ instruction is executed, IOCS searches the previously specified track (stored in the 8-byte track-reference field) for the desired key.

Then when a DASD record containing the specified key is found, the data area of the record is transferred to the main-storage input area.

Only the specified track is searched unless the programmer requests that multiple tracks be searched on each READ instruction. A search of multiple tracks is specified by including the DTFDA entry SRCHM in the file definition. With this entry, the specified track and all following tracks are searched until the desired record is found or the end of the cylinder is reached. The search of multiple tracks continues through the cylinder even though part of the cylinder may be assigned to a different logical file.

Record Reference by ID

If the record reference is by ID (identifier in the count area of records), the second parameter in the READ instruction must be the letters ID, and the DTFDA entry READID must be included in the file definition.

Whenever this method of reference is used, the problem program must supply both the track information and the record number in the 8-byte track-reference field. When the READ instruction is executed, IOCS searches the specified track for the particular record. When a record containing the

specified ID is found, both the key area (if present and specified in DTFDA KEYLEN) and the data area of the record are transferred to the main-storage input area.

WRITE MACRO

Name	Operation	Operand
[name]	WRITE	{Filename / (1)}, {KEY / ID / AFTER[,EOF] / RZERO}

The KEY, ID, or AFTER form of the instruction causes a record, which has been built in an output area of main storage, to be transferred from main storage to DASD storage. The output area must be specified in the DTFDA entry IOAREA1.

The first parameter specifies the symbolic name of the file to which the record is to be transferred. This name is the same as the one specified in the DTFDA header entry for this file and can be given as a symbol or in register notation. The second parameter specifies the type of reference that is used for searching the records on DASD to find the proper location to write the output record. The third parameter is optional and applies only to the WRITE Filename,AFTER form of the macro instruction.

The WRITE Filename,AFTER,EOF form of the macro instruction is used to write an end-of-file record (a record with a length of zero) on a specified track after the last record on this track.

The instruction WRITE Filename,RZERO is used to reset the capacity record of a specified track to its maximum value and to erase this track after record zero.

If records in the file are specified as undefined (RECFORM=UNDEF), the programmer must determine the length of each record and load it in a register for IOCS use before he issues the WRITE instructions for that record. The register that will be used for this purpose must be specified in the DTFDA entry RECSIZE.

Record Reference by Key

If the DASD storage location for writing records is determined by the record key (control information in the key area of the DASD record), the word KEY is entered as the second parameter of the WRITE macro instruction. Also the DTFDA entry WRITEKY must be included in the file definition.

Whenever this method of reference is used, the problem program must supply the key of the desired record to IOCS before the WRITE instruction is issued. For this, the key must be stored in the key field (specified by the DTFDA entry KEYARG). When the WRITE instruction is executed, IOCS searches the previously specified track (stored in the 8-byte track-reference field) for the desired key. Then, when a DASD record containing the specified key is found, the data in the main storage output area is transferred to the data area of the DASD record. This replaces the information previously recorded in the data area. The DASD count field of the original record controls the writing of the new record. If a record is shorter than the original record, it is padded with zeros. A record longer than the original record is written only to the extent of the area indicated in the count field on the track, and any excess bytes are lost. In either case (short or long records) IOCS turns on the wrong-length-record bit in the error-status field.

Only the specified track is searched unless the programmer requests that multiple tracks be searched on each WRITE instruction. Searching multiple tracks is specified by including the DTFDA entry SRCHM in the file definition. In this case, the specified track and all following tracks are searched until the desired record is found or the end of the cylinder is reached. The search of multiple tracks continues through the cylinder even though part of the cylinder may be assigned to a different logical file.

Record Reference by ID

If the DASD storage location for writing records is determined by the record ID (identifier in the count area of records), the letters ID are entered as the second parameter of the WRITE instruction. Also the DTFDA entry WRITEID must be included in the file definition.

Whenever this method of reference is used, the problem program must supply both the track information and the record number in the 8-byte track-reference field. When the WRITE instruction is executed, IOCS searches the specified track for the particular record. When the DASD record containing the specified ID is found, the information in the main storage output area is transferred to the key area (if present and specified in DTFDA KEYLEN) and the data

area of the DASD record. This replaces the key and data previously recorded. IOCS uses the count field of the original record to control the writing of the new record. If a record is shorter than the original record, it is padded with zeros. A record longer than the original record is written only to the extent of the area indicated in the count field on the track, and any excess bytes are lost. In either case (short or long records) IOCS turns on the wrong-length-record bit in the error/status field.

Record Reference: After

If a record is to be written following the last record previously written on a track (regardless of its key or ID), the second parameter of the WRITE instruction must be the specification AFTER. For this operation the DTFDA entry AFTER must be included in the file definition.

Whenever this method of reference is used for writing records, the problem program must supply the track information in the first seven bytes of the 8-byte track-reference field. When WRITE is executed, IOCS examines the capacity record (Record 0) on the specified track to determine the location and amount of space available for the record. If the remaining space is large enough, the information in the main-storage output area is transferred to the track in the location immediately following the last record. The count area, the key area (if present and specified by DTFDA KEYLEN), and the data area are written. IOCS then updates the capacity record.

If the space remaining on the track is not large enough for the record, IOCS does not write the record and, instead, sets an indication in the user's error/status byte (specified by the DTFDA entry ERRBYTE).

Whenever a new file is built in an area of the disk pack or cell that contains outdated records, the capacity records must first be set up to reflect empty tracks. An IBM-supplied utility program (Clear Disk) is available to construct Record 0.

Record Reference: RZERO

The RZERO instruction resets the capacity record to reflect an empty track. The problem program must supply, in SEEKADR, the cylinder and track number of the track to be reinitialized. Any record number is valid but will be ignored. IOCS will write a new R0 with the maximum capacity of the track (3625 for an IBM 2311; 2000 for an IBM 2321) and will erase the full track after R0.

This macro should be used every time the problem program wants to reuse a certain portion of a pack. It may be used as a utility function to initialize a limited number of tracks or cylinders.

WAITF MACRO

Name	Operation	Operand
[name]	WAITF	{Filename / (1)}

The WAITF macro instruction is used to ensure that the transfer of a record has been completed. It requires only one parameter: the name of the file containing the record. The parameter can be specified as a symbol or in register notation.

This instruction must be issued before the problem program attempts to process an input record or build another output record for the file concerned. The program does not regain control until the transfer of data is complete. Thus, the WAITF macro instruction must be issued after any READ or WRITE instruction for a file, and before the succeeding READ or WRITE instruction for the same file.

The WAITF macro makes error/status information, if any, available to the problem program in the field specified by DTFDA ERRBYTE.

CNTRL MACRO

Name	Operation	Operand
[name]	CNTRL	{Filename / (1)},code

The CNTRL (control) macro instruction can be used to begin access movement for the next READ or WRITE for a file. It requires two parameters.

The first parameter specifies the name of the file, which is the same name as that specified in the DTFDA header entry for the file, and can be given as a symbol or in register notation. The second parameter must be the word SEEK (for 2311 and 23[illegible] or RESTR (for 2321 only). The seek address must be provided in the field with the symbolic name given in the DTFDA entry SEEKADR.

PROCESSING DASD RECORDS BY THE INDEXED SEQUENTIAL SYSTEM

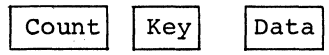
The Indexed Sequential File Management System (ISFMS) permits DASD records to be processed in random order or in sequential order by control information. For random processing, the user supplies the key (control information) of the desired record to ISFMS and issues a READ or WRITE macro instruction to transfer the specified record. For sequential processing by control information (key), the user specifies the first record to be processed and then issues GET or PUT macro instructions until all desired sequential records have been processed. The successive records are made available in sequential order by key. Variations in macro instructions permit:

- A logical file of records to be loaded onto DASD (created).
- Individual records to be read from, added to, or updated in the file.

Whenever the indexed sequential system of processing is used, the logical file and main-storage areas allotted to the file must be defined by the declarative macro DTFIS (Define The File for Indexed Sequential System). The detail parameter entries for this definition are described under Declarative Macro Instructions.

RECORD TYPES

When an ISFMS file is originally organized, it is loaded onto the volume(s) from presorted input records. These records must have been sorted by control information. All records in the file must contain key areas:



All keys must be the same length, and the length must be specified in the DTFIS entry KEYLEN.

The logical records must be fixed length, and the length must be specified in the DTFIS entry RECSIZE. Logical records may be either blocked (two or more logical records in one physical record) or unblocked (one logical record per one physical record). The number specified in the [illegible] than blocked records [illegible] the highest record key for the [illegible] stored in the [illegible] number of [illegible] specified in the [illegible]

STORAGE AREAS

Records in one logical file are transferred to, or from, one or more I/O areas in main storage. The areas must always be large enough to contain the key area and a block of records, or a single record if unblocked records are specified. In addition, it must allow space for the count area when a file is to be loaded, or when records are to be added to a file. For the functions of adding or retrieving records, the I/O area must also provide space for a sequence-link field that is used in conjunction with overflow records (see Addition of Records and Overflow Areas). The I/O area requirements are illustrated schematically in Figure 10 and described in detail in the DTFIS entries IOAREAL, IOAREAR, and IOAREAS.

Records may be processed directly in the I/O area or in a work area. If the records are to be processed in the I/O area, a register must be specified in the DTFIS entry IOREG. This is used for indexing, to point to the beginning of each record and thus locate the record for processing.

If the records are to be processed in a work area, the DTFIS entry WORKL, WORKR, or WORKS must be specified. ISFMS moves each individual input record from the I/O area to the work area where it is available to the problem program for processing. Similarly, on output ISFMS moves the completed record from the work area to the I/O area where it is available for transfer to DASD storage. Whenever a work area is used, a register is not required.

ORGANIZATION OF RECORDS ON DASD

When a logical file of presorted records is loaded into DASD, ISFMS organizes the file in a way that allows the user to have access to any record, in the most efficient manner.

Reference can be made to records at random throughout the logical file, or to a series of records in the file in their presorted sequence (collating sequence). The organization also provides for additions to the file at a later time, while still maintaining both the random and sequential reference capabilities.

ISFMS loads the records, one after the other, into a specified area of the DASD volume. This is called the prime area of the logical file on DASD. The starting and ending limits of this area are specified by the user in Job Control XTENT cards.

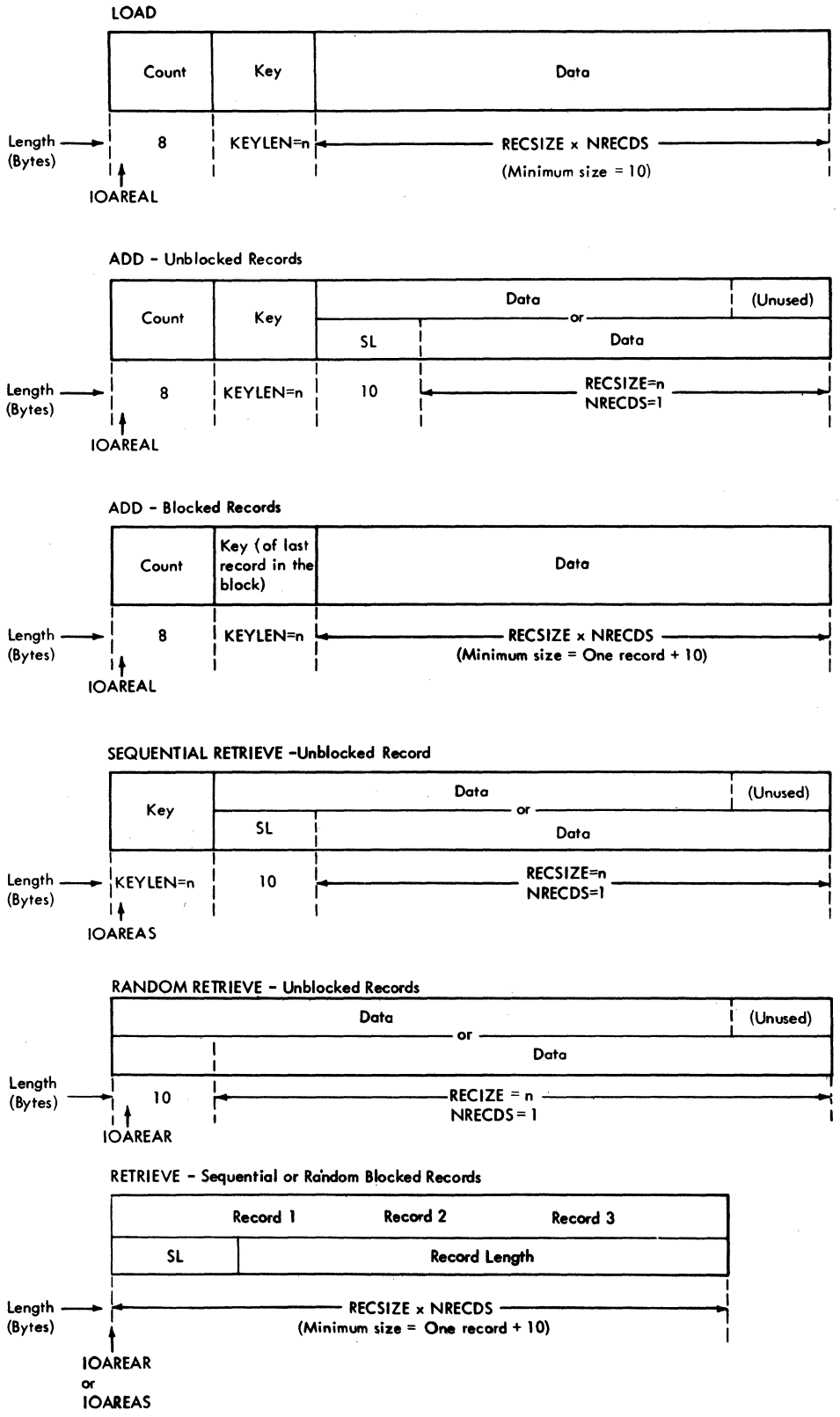


Figure 10. Schematic of I/O Areas in Main Storage, for ISFMS

Indices

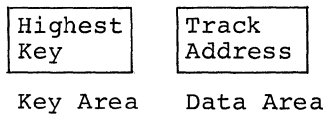
As ISFMS loads a file of records sorted by control information, it builds a set of indices for the file. The indices are utilized for both random and sequential reference to records as follows:

- They permit rapid access to individual records for random processing.
- They supply the means of providing records in key order during sequential processing.

Either two or three indices are built, depending on the user's specifications. Both a track index and a cylinder index are always constructed. A master index is also constructed if the DTFIS entry MSTIND is included in the file definition.

Once a file has been loaded and the related indices have been built, the ISFMS routines search for specified records by referring to the indices. When a particular record (specified by key) is requested for processing, ISFMS searches the master index (if used), then the cylinder index, then the track index, and finally the individual track. Each index narrows the search by pointing to the portion of the next-lower index whose range includes the specified key. Because of the high speed and efficiency of the direct access devices in a System/360, a master index should be established only for exceptionally large files, for which the cylinder index occupies several tracks (possibly five or more). That is, it is generally faster to search only the cylinder index (followed by the track index) when the cylinder index occupies four or less tracks.

The indices are made up of a series of entries, each of which includes the address of a track and the highest key on that track or cylinder. Each entry is a separate record composed of both a key area and a data area. The key area contains the highest key on the track or cylinder, and its length is the same as that specified for logical data records (in the DTFIS entry KEYLEN). The data area of each index is ten bytes long, and it contains track information including the track address.



Track Index

The track index is the lowest-level index for the logical file. A separate track index is built for each cylinder used by the file, and it contains index entries for that cylinder only. Each track index is located on the cylinder that it is indexing. It is always on the first track of that cylinder.

When the track indices are originally constructed, they contain two similar entries (normal and overflow) for each track utilized on the cylinder. For example, if the prime area of the logical file utilizes eight tracks on a cylinder, the track index might contain the entries shown in Figure 11. The use of two index records for each track is required because of overflow records that will occur if more records are inserted in the file at a later

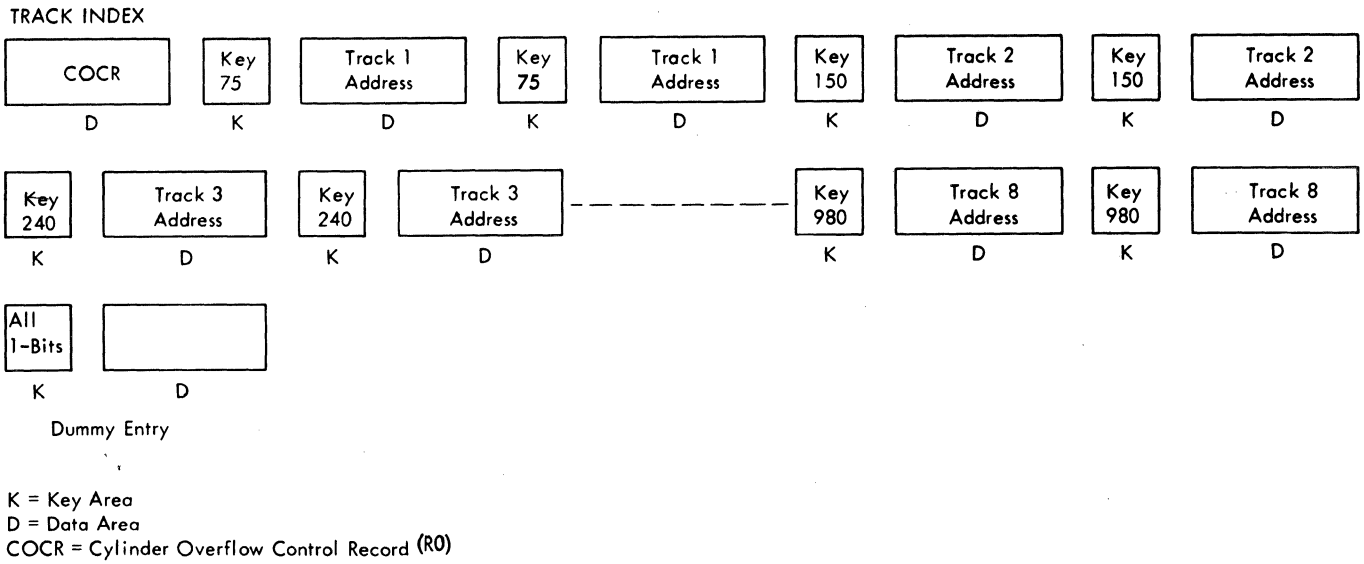


Figure 11. Schematic Example of a Track Index

time (see Addition of Records and Overflow Areas). When overflow records for a track exist, the second (overflow) index record contains the key of the highest record in the overflow chain and the address of the lowest record in the overflow chain for the track. The dummy entry indicates the end of the track index. Any following records are logical-file data records.

Cylinder Index

The bar-position address of a 2321 data cell corresponds to the cylinder of a 2311 disk drive in ISFMS. References to cylinder also apply to the 2321. The cylinder index is an intermediate level index for the logical file. It contains an index entry for each cylinder occupied by the file. This index is built in the location specified by the user in a Job Control XTENT card. The cylinder index may be built wherever the user chooses, but it may not be on one of the cylinders that contains data records for this file. It must be on a separate cylinder, or it may be on a separate volume that will be on-line whenever this logical file is processed.

The cylinder index may be located on one or more successive cylinders. Whenever the index is continued from one cylinder to another, the last index entry on the first cylinder contains a linkage field that points to the first track of the next cylinder. A cylinder index may not be continued from one volume to another, however. It must be completely contained within one volume.

This index contains one entry for each cylinder occupied by the data file. The key area contains the highest key associated with the cylinder, and the data area contains the address of the track index for that cylinder. For example if a file requires nine cylinders, the cylinder index might contain the entries shown in Figure 12. The dummy entry indicates the end of the cylinder index.

Master Index

The master index is the highest-level index for a logical file built by the IBM System/360 Disk Operating System. This index is optional, and it is built only if it is specified by the DTFIS entry MSTIND. It is built in the location specified by a Job Control XTENT card. Like the cylinder index, it may be located on the same volume with the logical-file records or on a different volume that will be on-line whenever the records are processed.

The master index must immediately precede the cylinder index on a volume, and it may be located on one or more successive cylinders. Whenever it is continued from one cylinder to another, the last index entry on the first cylinder contains a linkage field that points to the first track of the next cylinder. A master index may not be continued from one volume to another, however. It must be completely contained within one volume.

The master index contains an entry for each track of the cylinder index. The key area contains the highest key on the cylinder index track, and the data area contains the address of that track. For example, if a master index is located on track x and a cylinder index is located on tracks x+1 through x+20, the master index might contain the entries shown in Figure 13. The dummy entry indicates the end of the master index.

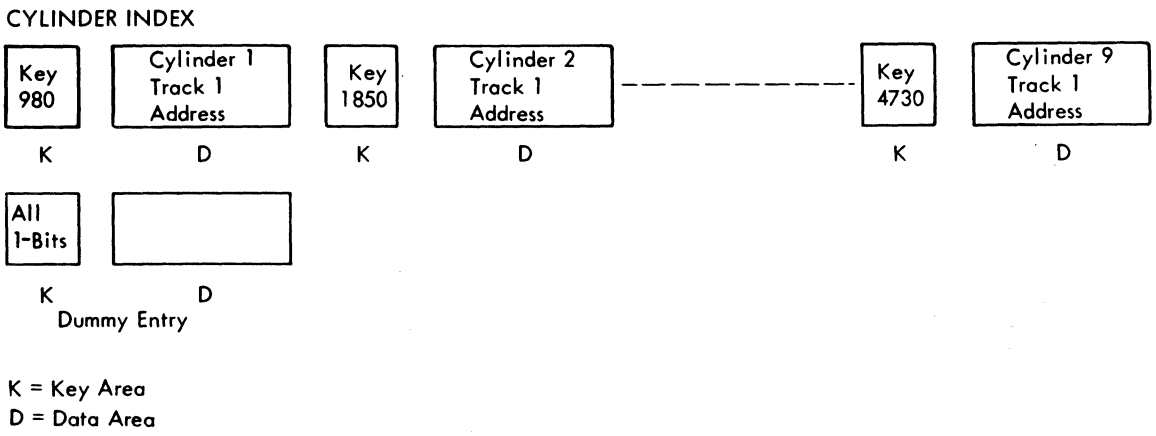


Figure 12. Schematic Example of a Cylinder Index

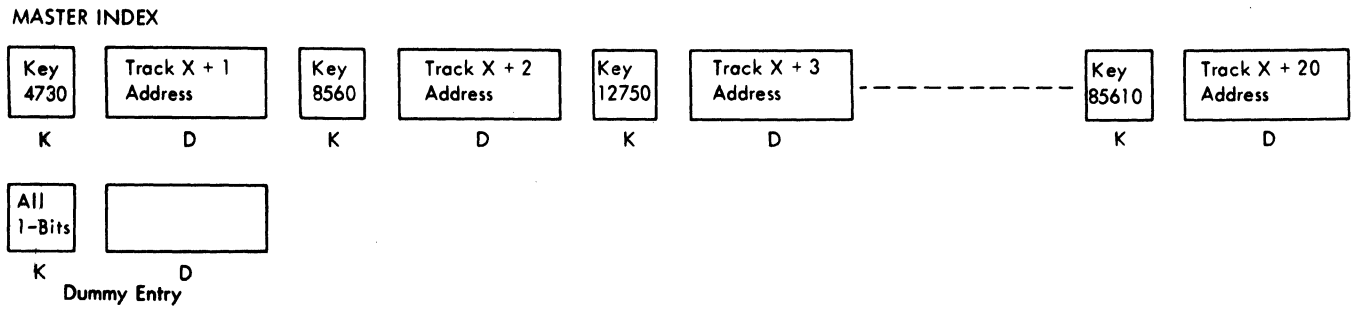


Figure 13. Schematic Example of a Master Index

ADDITION OF RECORDS AND OVERFLOW AREAS

After a logical file has been organized on DASD, it may subsequently become necessary to add records to the file. These records may contain keys that are above the highest key presently in the file and, thus, constitute an extension of the file. Or, they may contain keys that fall between keys already in the file and therefore require insertion in the proper sequence in the organized file.

If all records to be added have keys that are higher than the highest key in the organized file, the upper limit of the prime area of the file can be adjusted (if necessary) by the specification in a Job Control XTENT card, and the new records can be added by presorting them and loading them into the file. No overflow area is required. The file is merely extended further on the volume. However, new records can be batched with the normal additions and added to the end of the file.

If records must be inserted among those already organized, however, an overflow area will be required. ISFMS uses the overflow area to permit the insertion of records without necessitating a complete reorganization of the established file. The fast random and sequential retrieval of records is maintained by inserting references to the overflow chains in the track indices, and by using a chaining technique in the overflow records. For chaining, a sequence-link field is prefixed to the user's data record in the overflow area. The sequence-link field contains the address of the record in the overflow area that has the next-higher key. Thus a chain of sequential records can be followed in a search for a particular record. The sequence-link field of the highest record in the chain indicates the end of the chain. All records in the overflow area are unblocked, regardless of the specification (in DTFIS RECFORM) for the data records in the logical file.

To add a record by insertion, ISFMS searches the established indices first to determine on which track the record must be inserted. The keys of the last records on the tracks in the originally organized file determine the track where an inserted record belongs. A record is always inserted on the track where:

1. The last key is higher than the insertion, and

2. The last key of the preceding track is lower than the insertion.

For example, assume Tracks 2 and 3 are organized with the record keys shown in Figure 14. Then records with keys such as 151, 175, 199, 215, and 239 are inserted on Track 3 (or in the related overflow chain that has developed). Any key lower than 150 is added to either Track 1 or Track 2; any key higher than 240 belongs to Track 4 or above. The track indices always retain the highest key of each track as it was originally organized.

After the proper track is determined, ISFMS searches the individual records on the track or overflow area (if necessary) to find where the record belongs in key order. This results in either of two conditions:

1. The record falls between two records presently on the track. ISFMS adds the record by inserting it in the proper sequence and shifting each succeeding record one record location higher on the track, until the end record is forced off the track. ISFMS transfers the end record to the overflow area,

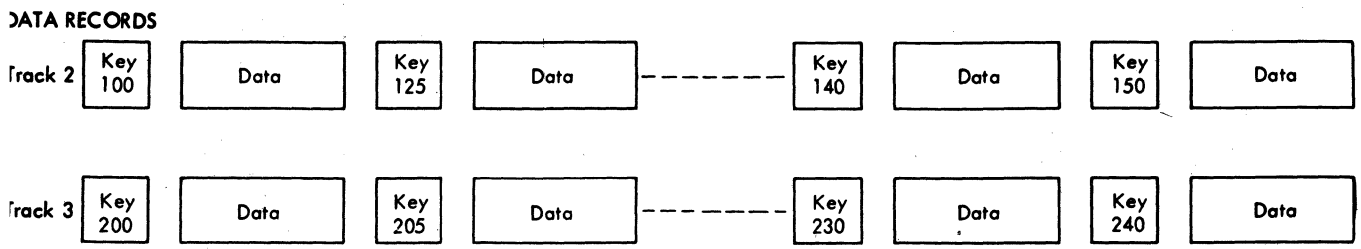


Figure 14. Example of Data Records as Originally Organized on Tracks 2 and 3

and prefixes the record (data area) with a sequence-link field. The first time a record is inserted on a track, the sequence-link of the overflow record indicates that this is the highest record associated with the track. Thereafter, the sequence-link field of each overflow record points to the next-higher record for that track.

ISFMS also updates the track index to reflect this change. The first index record for the track has the key field changed to indicate the new last-record located on the track. The second index record for the track has the track address (in the data area) changed to point to the address of the overflow record. If a record with key 105 is added to a file organized as shown in the previous illustrations and if the overflow area is located on Track 9, the track index records contain the information shown in Figure 15.

2. The record falls between the last record presently on the track and the last record originally on the track. Thus, it belongs in the overflow area. ISFMS writes the record in the overflow area following the last record previously written. ISFMS searches through the chain of records associated with the corresponding track for this record and identifies the sequential position the record should take. Then the sequence-link fields of the new record, and of the record preceding it by sequential key, are adjusted to point to the proper records. If records 150, 140, and 130 are already in the overflow area and record 135 is to be added, for example, the sequence-link fields of records 130 and 135 must be adjusted (Figure 16).

Overflow-Area Option: The location of the overflow area(s) for a logical file may be specified by the user. The overflow areas may be built by one of three methods:

1. Overflow areas for records may be located on each cylinder within the prime area that is specified by a Job Control XTENT card for the data file. In this case the user must specify the number of tracks to be reserved for overflow on each cylinder occupied by the file. The overflow records that occur within a particular cylinder are written in the cylinder overflow area for that cylinder.

   The number of tracks to be reserved for each cylinder overflow area must be specified in the DTFIS entry CYLOFL when a file of records is to be loaded and when records are to be added to an organized file.

2. An independent overflow area may be specified for storing all overflow records for the logical file. In this case a Job Control XTENT card must be included when the program is executed to specify the area of the volume to be used for the overflow area. This area may be on the same volume with the data records, or on a different volume that is on-line. However, it must be contained within one volume. (It must be the same kind of device as that containing the prime data area.)

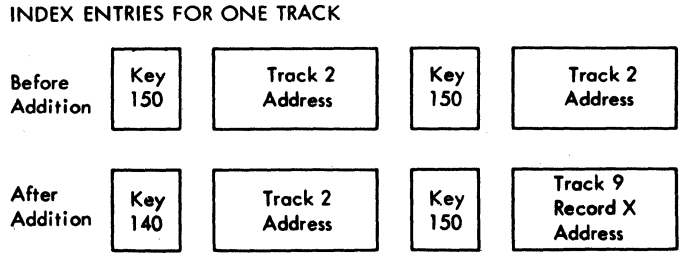


Figure 15. Example of Track Index Entries Before and After Addition of a Record on Track 2

RECORD	SEQUENCE-LINK FIELD	
	Before Addition	After Addition
130	140	135
135 (New Record)	—	140

Figure 16. Example of Sequence Link Fields Adjusted for Addition of a Record 135

3. Both cylinder overflow areas (method 1) and an independent overflow area (method 2) may be used. In this case overflow records are placed first in the cylinder overflow areas within the data file. When any cylinder overflow area becomes filled, the additional overflow records from that cylinder are written in the independent overflow area. The specifications required for both methods 1 and 2 must be included for this combined method of handling overflows.

All records placed in the overflow area will be in the unblocked format and will have prefixed to each record a sequence-link field. There must always be one prime data track available (for a DASD record which has a data length of zero) when additions are being made to the last track in the prime data area containing records.

EXAMPLE OF AN ORGANIZED FILE

A simplified example of a file organized on DASD by the Indexed Sequential File Management System is shown schematically in Figure 17. Figure 17 is an illustration of an organized file for an IBM 2311 DASD with the last two tracks on each cylinder used for the overflow area. The same file would have similar characteristics if it was created on another IBM DASD type. The assumptions made and the items to be noted are:

1. The track index occupies part of the first track, and data records occupy the rest of the track.

2. The data records occupy part of Track 0 and all of Tracks 1-7. Tracks 8 and 9 are used for overflow records in this cylinder.

3. The master index is located on Track X on a different cylinder. The cylinder index is located on Tracks X+1 through X+20.

4. A dummy entry signals the end of each index.

5. The file was originally organized with records as follows:

Track	Records
1	5-75
2	100-150
3	.
.	.
.	.
8	900-980

6. The track index originally had two entries for each track. It now shows that overflow records have occurred for Tracks 1 and 7.

7. Records 150, 140, and 130 were forced off the track by insertions on the track. Record 135 was added directly in the overflow area.

8. A sequence-link field (SL) has been prefixed to each overflow record. The records for Track 1 can be searched in sequential order by following the SL fields:

Record	Sequence-Link Field (SL)
130	SL points to record with key 135.
135	SL points to record with key 140.
140	SL points to record with key 150.
150	End of search. (Key 150 was the highest key on Track 2 when the file was loaded.)

9. When the file was loaded, the last record on Cylinder 1 was Record 980, on Cylinder 2 Record 1850, and on Cylinder 9 Record 4730. This is reflected in the cylinder index. The first entry in the master index is the last entry of the first track of the cylinder index.

10. When cylinder overflow areas are used, the first record (Record 0) in the track index for a cylinder is the Cylinder Overflow Control Record (COCR) It contains the address of the last overflow record on the cylinder and the number of tracks remaining in the cylinder overflow area. When the number of remaining tracks is zero, overflow records are written in the independent area. The format of record zero data field is as follows:
HHRBBTXX

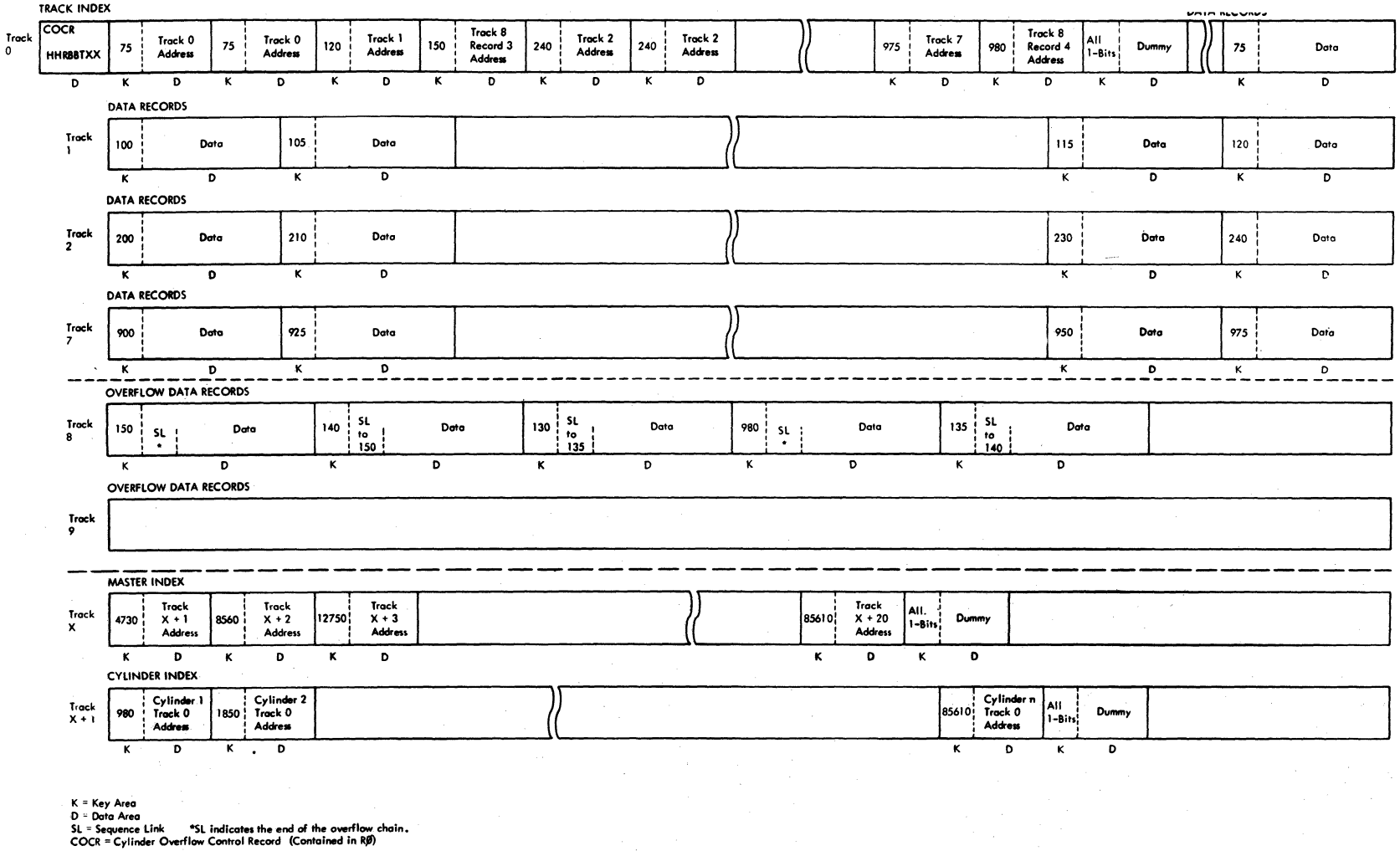
HH - last cylinder overflow track containing the records.

R - last overflow record on the track.

BB - the number of bytes remaining on the track (for fixed-length record this will be binary zeros).

T - the number of remaining tracks available in the cylinder overflow area.

XX - reserved (with binary zeros).



K = Key Area
D = Data Area
SL = Sequence Link *SL indicates the end of the overflow chain.
COCR = Cylinder Overflow Control Record (Contained in RØ)

Figure 17. Schematic of a File on 2311 DASD Organized by ISFMS

MACRO INSTRUCTIONS TO LOAD OR EXTEND A DASD FILE BY ISFMS

The function of originally loading a file of presorted records onto DASD, and the function of extending the file by adding new presorted records beyond the previous high record, are the same. Both are considered a LOAD operation (specified by the DTFIS entry IOROUT), and they both use the same macro instructions in the problem program. However, the type field in the DLAB card must specify ISC for load creation and ISE for load extension.

The areas of the volumes used for the file are specified by Job Control XTENT cards. The areas are: the prime area where the data records are written, a cylinder index area where the user wants ISFMS to build the cylinder index, and a master index area if a master index is to be built (specified by the DTFIS entry MSTIND).

During the load operation, ISFMS builds the track, cylinder, and master (if specified) indices.

Three different macro instructions are always required in the problem program to load original or extension records into the logical file on DASD.

SETFL MACRO

Name	Operation	Operand
[name]	SETFL	{Filename / (0)}

The SETFL (set file load mode) macro instruction causes ISFMS to set up the file so that the load function can be performed. SETFL preformats the last track index of each cylinder of a file with zero entries. The name of the file to be loaded is the only parameter required in this instruction. This name is the same as that specified in the DTFIS header entry for this file. It can be specified as a symbol or in register notation.

This macro must be issued whenever the file is to be loaded or extended.

WRITE MACRO

Name	Operation	Operand
[name]	WRITE	{Filename / (1)} ,NEWKEY

When a WRITE macro instruction with the parameter NEWKEY is issued in the problem program between a SETFL instruction and an ENDFL instruction (the third macro required for loading), it causes ISFMS to load a record onto DASD.

It requires two parameters. The first specifies the name of the file specified in the DTFIS header entry. The filename can be specified as a symbol or in register notation. The second parameter must be the word NEWKEY.

Before issuing the WRITE instruction, the problem program must store the key and data portions of the record in a work area (specified by DTFIS WORKL). The ISFMS routines construct the I/O area (see Figure 10) by moving the data record to the data area, moving the key to the key area, and building the count area. When the I/O area has been filled, ISFMS transfers the records to DASD storage and then constructs the count area for the next record. The WAITF macro should not be used when loading or extending an ISFMS file.

Before records are transferred, ISFMS performs both a sequence check (to ensure that the records are in order by key) and a duplicate-record check.

After each WRITE is issued, ISFMS makes the ID of that record or block available to the problem program. The ID is located in an 8-byte field labeled FilenameH. (Filename cannot exceed 7 characters.) For example, if the file name in the DTFIS header entry is PAYRD, the ID field is addressed by PAYRDH. By reference to this field, the ID of any selected records can be punched or printed for later use. This will be required if the user plans to retrieve records in sequential order starting with the ID of a particular record (see SETL Macro).

As records are loaded onto DASD, ISFMS writes track-index records each time a track is filled, writes a cylinder-index record each time a cylinder is filled, and writes a master-index record (if DTFIS MSTIND is specified) each time a track of the cylinder index is filled.

ENDFL MACRO

Name	Operation	Operand
[name]	ENDFL	{Filename / (0)}

The ENDFL (end file load mode) macro instruction ends the mode initiated by the SETFL macro. The name of the file that has been loaded is the only parameter required in this instruction. This name is the same as the name specified in the DTFIS header entry for this file. It can be specified as a symbol or in register notation.

The ENDFL macro performs a close-like operation for the file that has been loaded. It writes the last block of data records, if necessary, and then writes an end-of-file record after the last data record. It writes any index entries that are needed. It also writes dummy index entries for the unused portion of the prime data XTENT.

MACRO INSTRUCTIONS TO ADD RECORDS TO A DASD FILE BY ISFMS

After a file has been organized on DASD, new records can be added to the file. Each record is inserted in the proper place sequentially by key. This function is provided by specifying ADD or ADDRTR in the DTFIS entry IOROUT.

The file may contain either blocked or unblocked records, as specified by the DTFIS entry RECFORM. When the file contains blocked records, the user must provide ISFMS with the location of the key field that is provided through the DTFIS entry KEYLOC. The records to be inserted are written one record at a time. The records must contain a key field in the same location as the records already in the file. Whenever the addition of records is to follow sequential retrieval (ADDRTR) the macro instruction ESETL must be issued before a record is added.

Two macro instructions, WRITE and WAITF, are used in the problem program for adding records to a file.

WRITE MACRO

Name	Operation	Operand
[name]	WRITE	{Filename / (1)},NEWKEY

The operand Filename is the same name that is contained in the DTFIS header entry. The name can be specified as a symbol or in register notation.

Before the WRITE macro is issued for unblocked records, the program must store the record (key and data) to be added into a work area specified in the DTFIS entry WORKL. For blocked records, the program must store only the data (the key is assumed to be a part of the data). Before any records are transferred, ISFMS checks for duplicate record keys. If no duplication is found, ISFMS will insert the record in the file.

To insert a record into a file, ISFMS performs an index search at the highest level index. This search determines if the record to be inserted can be placed within the file, or if it is higher than the last record on the file.

If the record can be inserted within the file, searching of the master index (if available), the cylinder index, and the track index determines the appropriate location to insert the record.

For an entry to an unblocked file, an equal/high search is performed in the prime data area of the track. When a record on the track is found that is equal or higher than the record to be inserted, the record is read from the track and placed in main storage (in the I/O area). The two records are compared to see if a duplicate record is found. If a duplication is found, that information will be posted to the user in the DTF table at FilenameC. If no duplicate is found, the appropriate record (in the user's work area) is written directly to the track. The record (just displaced from the track) in the I/O area is moved by ISFMS to the user's work area.

The next record on the track is read into the I/O area. Then the record in the work area is written on the track. Succeeding records are shifted until the last record on the track is set up as an overflow record. This last record is then written

into the appropriate overflow area, and the appropriate track index entries are updated. This is the cylinder overflow area, if CYLOFL has been specified for this file and the area has not been filled.

If the cylinder overflow area is filled, or if only an independent area has been specified by a Job Control XTENT card, the end record is transferred to the independent overflow area. If an independent overflow area has not been specified (or is filled) and the cylinder area is filled, there is no room available to store the overflow record. ISFMS will post this condition in the DTF table at FilenameC.

In all cases, before any records are written, ISFMS determines if room is available.

For an entry to a blocked file, the work area, WORKL, is required in the DTFIS entries. Each record to be added must contain a key field in the same location as the records already in the file. The high-order position of this key field, relative to the leftmost position of the logical record, must be specified to ISFMS by the user. The DTFIS entry KEYLOC is used for this specification.

When the WRITE macro is issued in the problem program, ISFMS first locates the correct track by referring to the necessary master (if available), cylinder, and track indices. Then a search on the key areas of the DASD records on the track is made to locate the desired block of records. The block of records is read into the I/O area. ISFMS then examines the key fields within each logical record to find the exact position to insert the record. ISFMS checks for duplication of records. If an equal condition exists, the information is posted in FilenameC. If the record is high, the record in this position is exchanged with the record to be inserted (contained in the work area WORKL).

This procedure continues with each succeeding record in the block until the last record is moved into the work area. ISFMS then updates the key area of the DASD record and writes the block back onto DASD. The remaining blocks on the track are similarly processed until the last logical record on the track is moved into the work area. This record is then set up as an overflow record with the proper sequence-links, and moved to the overflow area. The indices are updated and ISFMS returns to the problem program for the next record to be added. If the overflow area is filled, the information will be posted in FilenameC.

If the proper track for a record is an overflow track (determined by the track index), ISFMS searches the overflow chain and checks for duplication. If no duplication is found, ISFMS writes the record, preceded by a sequence-link field in the data area of the DASD record, and adjusts the appropriate linkages to maintain sequential order by key. ISFMS writes the new record in either the cylinder overflow area or the independent overflow area. If these areas are filled, the user will be notified by a bit in FilenameC.

If the new record is higher than all records presently in the file (end-of-file), ISFMS checks to determine if the last track containing data records is filled. If it is not, the new record is added, replacing the end-of-file record. The end-of-file record is written in the next record location on the track, or on the next available prime data track. Another track must be available within the file limits. If the end-of-file record is the first record on a track, the new record is written in the appropriate overflow area. After each new record is inserted in its proper location, ISFMS adjusts all indices that are affected by the addition.

MACRO INSTRUCTIONS FOR RANDOM RETRIEVAL BY ISFMS

When a file has been organized by ISFMS, records can be retrieved in random order for processing and/or updating. Retrieval must be specified in the DTFIS entry IOROUT (IOROUT=RETRVE or IOROUT=ADDRTR). Random processing must be specified in the DTFIS entry TYPEFLE=RANDOM.

Because random reference to the file is by record key, the problem program must supply the key of the desired record to ISFMS. To do this, the key must be stored in the main storage key field specified by the DTFIS entry KEYARG. The specified key designates both the record to be retrieved and the record to be written back into the file in an updating operation.

Three macro instructions are available for use in the problem program for retrieving and updating records randomly.

READ MACRO

Name	Operation	Operand
[name]	READ	{Filename / (1)},KEY

The READ instruction causes ISFMS to retrieve the specified record from the file. This instruction requires two parameters. The first parameter specifies the name of the file from which the record is to be transferred to main storage. This name is the same as the name specified in the DTFIS header entry for this file and can be specified as a symbol or in register notation. The second parameter must be the word KEY.

To locate the record ISFMS searches the indices to determine the track on which the record is stored, and then searches the track for the specific record. When the record is found, ISFMS transfers it to the I/O area specified by the DTFIS entry IOAREAR. The ISFMS routines also move the record to the specified work area if the DTFIS entry WORKR is included in the file definition.

When records are blocked, ISFMS transfers the block that contains the specified record to the I/O area. It makes the individual record available for processing either in the I/O area or the work area (if specified). For processing in the I/O area, ISFMS supplies the address of the record in the register specified by DTFIS IOREG. The ID of the record can be referenced using FilenameG.

WRITE MACRO

Name	Operation	Operand
[name]	WRITE	{Filename / (1)},KEY

The WRITE instruction with the parameter KEY is used for random updating. It causes ISFMS to transfer the specified record from main storage to DASD storage. This instruction requires two parameters. The first parameter specifies the name of the file to which the record is to be transferred. This name is the same as the name specified in the DTFIS header entry and in the preceding READ instruction for this file. The name can be specified as a symbol or in register notation. The second parameter must be the word KEY.

ISFMS rewrites the record retrieved by the previous read instruction for the same file. The record is updated from the work area if one is specified, otherwise from the I/O area. The key need not be specified again ahead of the WRITE instruction.

WAITF MACRO

Name	Operation	Operand
[name]	WAITF	{Filename / (1)}

The WAITF macro instruction is issued to ensure that the transfer of a record has been completed. It requires only one parameter: the name of the file to which the record is being transferred. The name can be specified as a symbol or in register notation.

This instruction must be issued before the problem program attempts to process an input record or build another output record for the file concerned. The program does not regain control until the previous transfer of data is complete.

The WAITF instruction posts any exceptional information in the DTFIS table at FilenameC.

The WAITF instruction applies to the functions described in Macro Instructions to Add Records to a File by ISFMS, and Macro Instructions for Random Retrieval by ISFMS.

MACRO INSTRUCTIONS FOR SEQUENTIAL RETRIEVAL BY ISFMS

When a file has been organized by ISFMS, records can be retrieved in sequential order by key for processing and/or updating. The DTFIS entry IOROUT=RETRVE must be specified. Sequential processing must be specified in the DTFIS entry TYPEFLE=SEQNTL.

Although records are retrieved in order by key, sequential retrieval can start at a record in the file identified either by key or by the ID (identifier in the count area) of a record in the prime data area. Or, sequential retrieval can start at the beginning of the logical file. The user specifies, in SETL, the type of reference he will use in the problem program.

Whenever the starting reference is by key and the file contains blocked records (RECFORM=FIXBLK), the user must also provide ISFMS with the position of the key field within the records. This is specified in the DTFIS entry KEYLOC. To search for a record, ISFMS first locates the correct block by the key in the key area of the DASD record. (The key area contains the key of the highest record in the block.) Then, ISFMS examines the key field within each record in the block to find the specified record.

Four macro instructions are available for use in the problem program for retrieving and updating records sequentially.

SETL MACRO

Name	Operation	Operand
[name]	SETL	{Filename / (r)}, ({Idname / (r)}) {KEY / BOF / GKEY}

The SETL (set limits) macro instruction initiates the mode for sequential retrieval and initializes the ISFMS routines to begin retrieval at the specified starting address. It requires two parameters. The first operand (Filename) specifies the name of the file, specified in the DTFIS header entry, from which records are to be retrieved. The name can be given as a symbol or in register notation. Special register notation is not necessary to allow the macro to be used in a self-relocating program.

The second operand specifies where processing is to begin. If the user is processing by the record ID, the operand Idname or (r) specifies the symbolic name of the main-storage field in which the user will supply the starting (or lowest) reference for ISFMS use. The symbolic field will contain the following information:

Pointer to First Record to be Processed by Sequential Retrieval

Byte	Identifier	Contents	Information
0	M	2-245	Number of the extent in which the starting record is located.
1-2	B,B	0,0 (for 2311) 0, 0-9 (for 2321)	Always zero for 2311. Cell number for 2321.
3-4	C,C	0, 1-199 (for 2311) 0-19, 0-9 (for 2321)	Cylinder number for 2311. Sub-cell and strip for 2321. Note: The last four strips on each cell are reserved for alternate tracks.
5-6	H,H	0,0-9 (for 2311) 0-4,0-19 (for 2321)	Head position for 2311. Cylinder and head for 2321.
7	R	1-254	Record location.

If processing is to begin with a key supplied by the user, the second operand is KEY. The key will be supplied by the user in the field specified by the DTFIS entry KEYARG. If the specified key is not present in the file, an indication will be given at FilenameC.

The second operand BOF specifies that retrieval is to start at the beginning of the logical file.

Selected groups of records within a file containing identical characters or data in the first locations of each key can be processed by specifying GKEY in the second operand. The GKEY specification allows processing to begin at the first record (or key) within the desired group. The user must supply a key that will identify the significant (high order) bytes of the required group of keys. The remainder (or insignificant) bytes of the key must be padded with blanks, binary zeros, or bytes lower in collating sequence than any of the insignificant bytes in the first key of the group to be processed. For example, a GKEY specification of D6420000 would permit file processing to begin at the first record (or key) containing D642xxxx, regardless of the characters represented by the x's.

This method also allows starting at a key equal-to or greater-than the one specified in the DTFIS entry KEYARG without getting an error indication in FilenameC.

GET MACRO

Name	Operation	Operand
[name]	GET	{Filename / (1)} [,{Workname / (0)}]

The GET macro instruction causes ISFMS to retrieve the next record in sequence from the file. It can be written in either of two forms, depending on where the record is to be processed.

The first form is used if records are to be processed in the I/O area (specified by DTFIS IOAREAS). It requires only one parameter, which is the name of the file from which the record is to be retrieved. This is the same name as that specified in the DTFIS header entry. The name can be specified as a symbol or in register notation. ISFMS transfers the record from this file to the I/O area, and the record is available for the execution of the next instruction in the problem program. The key is located at the beginning of IOAREAS and the register (IOREG) points to the data. If blocked records are specified, ISFMS makes each record available by supplying its address in the register specified by the DTFIS entry IOREG. The key is contained in the record.

The second form of the GET instruction is used if records are to be processed in a work area (DTFIS specifies WORKS). It requires two parameters both of which can be specified as symbols or in register notation. The first is the name of the file, and the second is the name of the work area. When using register notation, workname should not be preloaded into register 1. The record is available for the execution of the next program instruction.

If blocked records are specified in the file definition, each GET that transfers a block of records to main storage will, if necessary, also write the preceding blank back into the file in its previous location. GET writes the preceding block if a PUT instruction has been issued for at least one of the records in the block. If no PUT instructions have been issued, updating is not required for this block and GET does not write the block. The ID of the record or block can be referenced by FilenameH.

PUT MACRO

Name	Operation	Operand
[name]	PUT	{Filename / (1)} [,{Workname / (0)}]

The PUT macro instruction is used for sequential updating of a file, and causes ISFMS to transfer records to the file in sequential order. PUT returns a record that was obtained by a GET. It may be written in either of two forms, depending on where records are processed.

The first form is used if records are processed in the I/O area (specified by DTFIS IOAREAS). It requires only the name of the file to which the records are to be transferred. This is the same as specified in the DTFIS header entry and can be specified in register notation or as a symbol.

The second form of the PUT instruction is used if records are processed in a work area. It requires two parameters, both of which can be specified either as a symbol or in register notation. The first is the name of the file, and the second is the name of the work area. When using register notation, workname should not be loaded into register 1. The work area name may be the same as that specified in the preceding GET for this file, but this is not

required. ISFMS moves the record from the work area specified in the PUT instruction to the I/O area specified for the file in the DTFIS entry IOAREAS.

When unblocked records are specified, each PUT writes a record back onto the file in the same location from which it was retrieved by the preceding GET for this file. Thus, each PUT updates the last record that was retrieved from the file. If some records do not require updating, a series of GET instructions can be issued without intervening PUT instructions. Therefore, it is not necessary to rewrite unchanged records.

When blocked records are specified, PUT instructions do not transfer records to the file. Instead, each PUT indicates that the block is to be written after all the records in the block have been processed. When processing for the block is complete and a GET is issued to read the next block into main storage, that GET also writes the completed block back into the file in its previous location. If a PUT instruction is not issued for any record in the block, GET does not write the completed block. The ESETL macro instruction writes the last block processed, if necessary before the end-of-file.

ESETL MACRO

Name	Operation	Operand
[name]	ESETL	{Filename (1)}

The ESETL (end set limit) macro instruction ends the sequential mode initiated by the SETL macro. The name of the file must be the same as the name specified in the DTFIS header entry. It can be specified as a symbol or in register notation.

If blocked records are specified, ESETL writes the last block back if a PUT was issued.

Note: If ADDRTR and RANSEQ are specified in the same DTF, ESETL should be issued before issuing a READ or WRITE, another SETL can be issued to restart sequential retrieval.

COMPLETION

After all the records for a logical output file have been processed (end-of-file), that file must be deactivated by an instruction in the problem program to close the file.

When the end of a logical input file in an I/O unit other than DASD or magnetic tape is sensed, IOCS immediately branches to the user's end-of-file routine (specified by EOFADDR) where the instruction to close the file can be issued.

When the end of a DASD or magnetic tape input file is sensed, IOCS checks standard trailer labels (if any), makes provision for user-checking of user labels, and then branches to the user's end-of-file routine (specified by EOFADDR) where the file may be closed. A CLOSE macro instruction is available to the programmer for closing each input and output file.

An end-of-volume condition (EOV), rather than an end-of-file condition (EOF), can occur during the processing of records in a logical file on DASD or magnetic tape. An EOV condition means that the processing of all the records on one volume has been completed, but that more records for the same logical file are recorded on another volume. When this occurs, IOCS checks or writes standard labels (if any) on the completed volume (trailer labels) and on the next volume (header labels), makes provision for user-processing of user-standard labels on both volumes, and then makes the data records on the next volume available for processing. Because IOCS detects the end-of-volume condition and utilizes many of the routines established for opening and closing files, no problem-program instructions are required specifically for an EOV condition. However, if the program requires that the processing of tape records on one volume be ended before the actual end-of-volume is reached, an end-of-volume condition can be forced. An FEOV (forced end-of-volume) macro instruction is provided for this condition in tape files.

The specific functions that occur on an EOF or EOV condition for a DASD or tape file vary with the type of operation (input or output) and with the use of file labels. These functions are discussed in the following sections.

DASD INPUT FILE

When records in a logical input file on DASD are processed in sequential order or in sequential order by key (specified by DTFIS or DTFSD), IOCS detects an end-of-file condition. The end of the input file is determined either by the ending address of the last extent specified for the file in Job Control XTENT cards, or by an end-of-file record read from the data file. With sequential processing of index sequential files (TYPEFLE=SEQNTL), IOCS posts an indication of this condition in bit 2 of the field referred to as FilenameC (see Figure 31 in the Indexed Sequential System (DTFIS) section of this publication). The user can test this bit and take any action required to close the file.

When records are processed in sequential order, the file may contain user trailer labels. In this case IOCS branches first to the user's label routine (specified by LABADDR) where the user may check his trailer labels. Up to eight trailer labels for a 2311 file and five trailer labels for a 2321 file can be read and checked. They are written on the first track of the first extent specified for the file on each pack. The trailer labels follow the additional user header labels for the pack, and they are identified by UTLx, where x is 1,2,... 5 or 1,2,...8. When IOCS branches to the user's label routine, it also reads the trailer label and makes it available to the user for checking. IOCS sets up a label area and supplies the address of the area to the user in Register 1. After each label is checked, the user returns to IOCS by use of the LBRET macro. After all trailer labels have been checked, IOCS branches to the user's end-of-file routine (specified by EOFADDR).

IOCS detects end-of-volume conditions in a DASD input file. The end of a volume is recognized when all extents on one volume have been processed but Job Control XTENT cards have specified additional extents on another volume. At the end of a volume, IOCS allows the user to check his trailer labels (if any), the same as at the end of a file. IOCS then checks the standard header labels on the next volume, allows the user to check any user header labels by branching to the address specified by LABADDR, and makes the first record in the first extent available for processing.

DASD OUTPUT FILE

When DASD records are processed sequentially or loaded sequentially by key (DTFIS), and when all records for the logical file have been completed, the CLOSE instruction is issued and normal EOF procedures are iniated (see CLOSE Macro). If the end of the last extent specified for the file is reached before CLOSE is issued, IOCS assumes an error condition.

End-of-volume conditions in a DASD output file are detected in the same way as in a DASD input file. At the end of a volume IOCS allows the user to write his trailer labels (if any), the same as at the end of a file (see CLOSE Macro). IOCS then writes standard file labels on the next volume, allows the user to write any user header labels by branching to the user's label routine (specified by LABADDR), and permits the processing of output data records to continue.

TAPE INPUT FILE

When logical IOCS senses a tape mark on a magnetic tape input file, either an end-of-file or end-of-volume condition exists. The EOF/EOV condition is determined by IOCS or by the user (depending on the type of labels used for the file) and the appropriate functions are performed.

If standard labels are specified, IOCS immediately reads and checks the standard trailer label. If user labels are also present and are to be checked (specified by LABADDR), the user's routine is then entered for each user label that is read (see OPEN Macro). Whenever LABADDR is specified in the DTF and multi-volume files are processed, the user's label routine must save and restore register 15 if any logical IOCS macros are used, other than LBRET. After all labels have been checked, the rewind option is executed, as specified in the DTF parameter REWIND.

When the standard trailer label is checked, either an EOV or EOF condition is sensed. When an EOV identifier is sensed, IOCS switches to the alternate tape drive (designated by ASSGN cards) after user labels have been checked if specified. If an alternate drive is not specified, the operator is notified to change the tape reels and the system enters the wait state. When the operator has mounted the new reel, processing resumes. IOCS checks the header label(s) if checking is specified, and normal processing continues. If an input file is processed by physical IOCS (DTFPH specified), the user must issue an OPEN instruction for the new reel. Then IOCS checks the header label and processing continues.

When an EOF condition is sensed, IOCS branches to the programmer's end-of-file routine, specified by the entry EOFADDR.

If the tape input file has nonstandard labels, IOCS immediately branches to the user's label routine (specified by LABADDR) when the tape mark is sensed. Whenever LABADDR is specified in the DTF and multi-volume files are processed, the user's label routine must save and restore register 15 if any logical IOCS macros are used, other than LBRET. In his routine, the programmer must use physical IOCS macro instructions to read his label(s). Furthermore he must determine the EOF/EOV condition and indicate this to IOCS by loading either EF (end-of-file) or EV (end-of-volume) in the two low-order bytes of Register 0. On an EOF condition, IOCS branches to the user's end-of-file address (specified by EOFADDR) when the problem program returns to IOCS at the end of the label routine. On an EV condition, IOCS initiates the end-of-volume procedures to close the completed volume and open the next volume for processing.

If a tape file is not labeled (FILABL=NO) or contains labels that are not to be checked (FILABL=NSTD) and no LABADDR entry, IOCS branches to the end-of-file address when the tape mark following the last data record is sensed.

Whenever an input tape is read backwards (READ=BACK), an end-of-file condition always exists when the file header label is reached. That is, backwards reading is confined to one volume. Therefore, with standard labels, the input/output routines check only the block count, which was stored from the trailer label, and then branch to the specified end-of-file routine. When physical IOCS macros are used to read records backwards, labels cannot be checked (DTFPH must not be specified). For tape files with nonstandard labels, IOCS branches to the user's label routine specified by LABADDR where he may check the header label. He must use physical IOCS macro instructions to read the label(s) for checking.

TAPE OUTPUT FILE

When an end-of-reel reflective marker is sensed on an output tape, logical IOCS prepares for closing the file by ensuring that all records have been written on the tape. If the programmer issues another PUT, indicating that more records are to be written on this output file, normal end-of-volume (EOV) procedures are initiated. If the programmer issues a CLOSE, the EOF procedures are initiated.

The programmer should be aware that, under certain conditions, an unfilled block of records may be written at an EOV or EOF condition, even though the file is defined as having fixed-length blocked records. When this file is used for input, the System/360 logical IOCS will recognize and handle these short blocks without the programmer being concerned or aware of this condition.

Labeling procedures for the EOV condition closely follow those described under CLOSE Macro. The label is coded EOV rather than EOF, and only one tape mark is written after the label set, or after the data if standard labels are not used.

FORCED END-OF-VOLUME: TAPE FILES

In some cases a programmer may need to force an end-of-volume condition at a point other than the normal tape mark (input) or reflective marker (output). He may want to discontinue reading or writing the records on the present volume and continue with those records for this same logical file that are recorded on the next volume. This may be necessary because of some major change in the category of records or in the processing requirements. An FEOV (forced end-of-volume) macro instruction is available to the programmer for this function. See FEOV Macro.

Op	Operand
for programs which are not self-relocating	
CLOSE	Filename1 (r1) [,{Filename2 (r2)},...,{Filenamen (rn)}]
for self-relocating programs	
CLOSER	Filename1 (r1) [,{Filename2 (r2)},...,{Filenamen (rn)}]

The CLOSE macro instruction is used to deactivate any file that was previously opened in any input/output unit in the system. (Console files cannot be closed.) A file may be closed at any time by issuing this macro instruction.

When the operation CLOSE is used, the symbolic address constants which CLOSE generates from the parameter list are not self-relocating. When CLOSER is specified, the symbolic address constants are self-relocating.

The symbolic name of the logical file (assigned in the DTF header entry) to be closed is entered in the operand field. A maximum of 16 files may be closed by one instruction, by entering additional filename parameters as operands. Alternately, the user can load the address of the filename in a register and specify the register using ordinary register notation. The address of the filename should not be preloaded into Register 0.

Reopening a DASD File

If further processing of a closed file is required at some later time in the program, the file must be opened again. If a file of DASD records is reopened after a CLOSE, the label processing and extents made available depend on the type of processing that is specified for this file. When an input file is processed in sequential order, IOCS checks the label(s) on the first volume and makes the first extent available, the same as at the original OPEN. When a file is processed by physical IOCS with SINGLE specified in the DTFPH parameter MOUNTED, IOCS opens the next extent specified by the user's XTENT cards. When a file is processed by the direct access method (DTFDA specified), by the indexed sequential system (DTFIS specified), or by physical IOCS with ALL specified in DTFPH MOUNTED, all label processing is repeated and all extents are again made available.

Reopening and Repositioning Tape Files

If further processing of a closed file is required at some later time in the program, the file must be opened again. After the CLOSE, the tape is positioned in accordance with the REWIND specification. Therefore to resume processing of tape records at the point where CLOSE occurred, NORWD should be specified in the DTF entry REWIND. When OPEN is issued later for additional records on that reel, the first record read must be a file label if standard labels are specified for the tape file being opened. If the tape file being opened is unlabeled or contains nonstandard labels, it is the user's responsibility to identify the first record read as a data record or a file label.

Closing DASD Files

When DASD records are processed in random order (specified by DTFDA or DTFIS), the CLOSE instruction is issued in the problem program to deactivate the file after all records have been processed.

When records in a DASD input file are processed in sequential order, the CLOSE instruction is generally issued in the user's end-of-file routine (specified by EOFADDR) to deactivate the file. IOCS branches to this routine when it detects an end-of-file condition (see Completion).

When records in a DASD output file are processed in sequential order or loaded in sequential order (DTFIS or DTFSD), the CLOSE instruction is issued after all records for the file have been processed. In this case CLOSE causes one or more functions to be performed before it deactivates the file. If records are processed in sequential order, user trailer labels may be written if the entry LABADDR is included in the file definition.

Up to eight trailer labels for a 2311 file and up to five for a 2321 file can be written on the first track of the first extent specified for the file on each volume. They follow the user header labels for the pack and are identified by UTLx. For this operation, IOCS branches to the user's label routine, sets up a label area, and supplies the address of the area in Register 1. In his routine the user constructs the trailer label in his own label area, places the address of this area for IOCS in Register 1, and then returns control to IOCS by use of the LBRET macro. IOCS then writes the trailer label. Similar to writing user header labels, these steps are repeated until eight (or five) trailer labels have been written or until the user indicates that he does not require any more labels, whichever occurs first. (See OPEN Macro: DASD Output File, Writing User-Standard Header Labels). After the last trailer label is written, CLOSE deactivates the file.

Closing Tape Input File

When an input file recorded on magnetic tape is processed, CLOSE is generally issued in the user's end-of-file routine. It initiates rewind procedures for the tape as specified in the DTF entry REWIND. It then deactivates the file.

Before CLOSE can be issued to an input file containing standard labels, all label processing and the rewind option must have been completed.

If CLOSE is issued for any tape input file before the end of the data is reached, the tape is rewound as specified by the entry REWIND, and the file is deactivated. No labels are read or checked.

Note: If CLOSE is issued to an input file that has not been opened, the option specified in the DTF entry REWIND will be performed.

Closing Tape Output File

For a magnetic tape output file, CLOSE is issued when all records for the file have been processed. It writes any record, or block of records, that has not already been written. If a record block is partially filled, it is truncated: that is, a short block is written on the tape. Following the last record, a tape mark is written. If labels are not specified, a second tape mark is written and the tape is rewound as specified in REWIND.

When standard labels are specified (STD in FILABL or OUTPUT in DTFPH TYPEFLE), CLOSE causes the file trailer label to be completely written after the tape mark. The EOF1 indication, the block count accumulated during the run, and the header-label information (with HDR1 replaced by EOF1) are included in the trailer label.

IOCS accumulates the block count for the trailer label whenever logical IOCS is used for an output file. When physical IOCS (DTFPH) is used, however, the problem program must accumulate the block count, if desired, and supply it to IOCS for inclusion in the standard trailer label. For this, the count (in binary form) must be moved to the 4-byte field within the DTF table labeled FilenameB. (Filename must be seven or less characters long.) For example, if the filename specified in the DTFPH header entry is DETLOUT, the block count field is addressed by DETLOUTB.

If checkpoint records are interspersed with data records on an output tape, the block count accumulated by logical IOCS does not include a count of the checkpoint records. Only data records are counted. Similarly if physical IOCS is used, the problem program must omit checkpoint records and count only data records.

If user labels are to follow the standard trailer, the CLOSE routine branches to the user's routine (identified by LABADDR) after the standard label has been written.

Whenever LABADDR is specified in the DTF and multi-volume files are processed, the user's label address routine must save and restore Register 15 if any logical IOCS macros are used other than LBRET. Upon entry to the user's routine, IOCS supplies Code F in the low-order byte of Register 0 to indicate that an end-of-file trailer label should be built. In his routine the programmer can build a maximum of eight labels, which the CLOSE routine writes for him. After building each user-standard label, he must return to the CLOSE routine by use of the LBRET macro.

After all trailer labels are written, the CLOSE routines write two tape marks, execute the rewind option, and deactivate the file.

Note: If CLOSE is issued to an output file that has not been opened, no tapemark or labels will be written.

For the proper procedures to handle user-standard labels and/or nonstandard labels, see OPEN Macro: Tape Output File.

Closing Other Files

When the last card, optical reader record, or paper tape input record has been read, IOCS branches to the user's end-of-file routine where CLOSE is generally issued.

When a printer or card output file is completed, CLOSE must be issued for that file. Any record in the output area that has not been printed or punched is transferred to the output file before the file is deactivated.

LBRET MACRO

Name	Operation	Operand
[name]	LBRET	{1 / 2}

The LBRET (label return) macro instruction is issued at the end of the user's label routine to return to IOCS. This macro is described under Initialization: LBRET Macro.

FEOV MACRO

Name	Operation	Operand
[name]	FEOV	{Filename / (1)}

The FEOV (forced end-of-volume) macro instruction is used for either input or output files on magnetic tape (programmer logical units only) to force an end-of-volume condition before sensing a tape mark or reflective marker. This indicates that processing of records on one volume is considered finished, but that more records for the same logical file are to be read from, or written on, the following volume. For system units see the SEOV macro.

For input magnetic tape, FEOV executes the rewind option selected by the user, provides for a reel change and processes the header label if it is specified in the DTF entry for the file. For an output file, the macro will write a tape mark. It also provides the ability to write the trailer label, provide a reel change, and process the next header label if it is specified in the DTF.

The name of the file, specified in the header entry, is the only parameter required in the operand. The name can be specified either as a symbol or in register notation.

When logical IOCS macro instructions are used for a file, FEOV initiates the same functions that occur at a normal end-of-volume condition, except trailer-label checking.

For an input tape, it immediately rewinds the tape as specified by REWIND and provides for a volume change as specified by the ASSGN cards. Trailer labels are not checked. FEOV then checks the standard header label on the new volume and provides for user-checking of any user-standard header labels if LABADDR is specified. If nonstandard labels are specified (FILABL=NSTD), FEOV provides for user-checking, if desired.

For an output tape, FEOV writes a tape mark. Then it writes the standard trailer label and user-standard labels (if any), writes one tape mark, provides for a volume change, and writes the file header label(s) on the new volume, as specified in the entries REWIND, FILABL, LABADDR, and the ASSGN cards. If nonstandard labels are specified, FEOV provides for user-writing of trailer labels (complete volume) and header labels (new volume), if desired.

When physical IOCS macro instructions are used and DTFPH is specified for standard label processing, FEOV may be issued for an output file only. In this case FEOV writes the standard trailer label, and any user-standard trailer labels if DTFPH LABADDR is specified. When the new volume is mounted and ready for writing, IOCS writes the standard header label and user-standard header labels, if any.

SEOV MACRO

Name	Operation	Operand
[name]	SEOV	Filename

The SEOV (system end-of-volume) macro instruction is used to cause automatic volume switching for magnetic tape output files if the physical end-of-volume is detected and SYSLST or SYSPCH are assigned to a tape output file. The routine will write a tape mark, rewind, unload the file, and check for an alternate tape. If none is found, a message is written and the user may mount a new tape on the same drive and continue. If an alternate unit is assigned, the macro fetches the alternate switching routine to promote the alternate unit, opens the new tape, and makes it ready for processing.

The two types of declarative macros, DTF's and Module Generation macros, are described in this section of the manual. The section is organized by type of processing:

- sequential
- direct access
- index sequential

Each type of processing is broken down further by type of file--card, magnetic tape, etc. The DTF used with the file is discussed first and then, where applicable, the module generation macro is discussed.

Note: The user does not need to specify names for his modules as illustrated by the examples in Appendix C. (None of those modules are user named.) If the user does not plan to name his modules, he can overlook the discussion on module-naming conventions following each module-generation macro instruction.

SEQUENTIAL PROCESSING

Eight DTF's can be used for sequential processing. The DTFCD, DTFMT, DTFPT, DTFOR, DTFCN, DTFPR, and DTFSD macros are subsets of the inclusive declarative macro, DTFSR. DTFSR is included for Basic Programming Support and Basic Operating System users. By specifying the subsets instead of DTFSR, program assembly time will be substantially improved.

CARD FILE (DTFCD)

Enter the symbolic name of the file, Filename, in the name field and DTFCD in the operation field. The detail entries follow the DTFCD header card in any order. Keyword entries are contained in the operand field. Figure 18 lists the entries.

BLKSIZE=n

Enter the length of the I/O area (IOAREA1) If the record format is variable or undefined, enter the length of the largest record. If this entry is omitted, the length is assumed to be 80.

CONTROL=YES

This entry is specified if a CNTRL macro is to be issued to the file. If this parameter is specified, CTLCHR must be omitted.

CRDERR=RETRY

This entry applies to card output on the IBM 2540 and 2520. It specifies the operation to be performed if an error is detected

If a punching error occurs, it usually is ignored and operation continues. The error card is stacked in pocket P1 (punch) and correct cards are stacked in the pocket selected by the user. If the CRDERR=RETRY entry is included and an error condition occurs, IOCS also notifies the operator and then enters the wait state. The operator can, by his reply to an appropriate message either terminate the job, ignore the error, or instruct IOCS to repunch the card. From this specification, IOCS generates a retry routine and a save area for the card punch record.

CTLCHR={ASA / YES}

This entry is required if first-character control is to be used. The ASA denotes the American Standard Association set. The YES denotes the System/360 character set. Appendix B contains the codes. This entry does not apply to combined files. If this parameter is specified, CONTROL must be omitted.

DEVADDR=SYSnnn

This entry specifies the symbolic unit (SYSnnn) to be associated with this logical file. An actual unit and channel will be assigned to the unit by an ASSGN card in the Job Control statement.

DEVICE=

This entry specifies the I/O device associated with this logical file. The acceptable entries are 2540, 1442, 2501, or 2520. If this entry is omitted, 2540 is assumed.

OPERATION	OPERAND	MUST BE INCLUDED	REMARKS
DTFCD		For each file	Header card. Specify 7-character or less symbolic filename in name field.
	BLKSIZE=n	For each file	Length of I/O area(s)
	CONTROL= YES	If a CNTRL macro is issued to the file	Omit CTLCHR operand.
	CRDERR= RETRY	To RETRY card output error	For 2520 and 2540 only
	CTLCHR=YES ASA	For first-character control	YES for S/360 character set. ASA for American Standard Set.
	DEVADDR= SYSnnn	For each file	Specifies symbolic unit
	DEVICE= 2540 1442 2520 2501	For device other than 2540.	To indicate I/O device
	EOFADDR= name	For input or combined file	Specifies end-of-file routine
	IOAREA1= name	For each file	Name=address expression
	IOAREA2= name	For combined file output area	Second output area Name=address expression
	IOREG=(r)	If two I/O areas are used with no work area.	Specify r (register 2-12)
	MODNAME= name	If name other than standard or a more inclusive module is referenced	Specifies a user-named I/O module
	OUBLKSZ=n	For a combined file if IOAREA2 is specified.	n=maximum number of characters to be transferred at one time
	RECFORM= FIXUNB UNDEF VARUNB	If other than FIXUNB.	Specifies record format Only FIXUNB valid for input.
	RECSIZE=(r)	For undefined records	For undefined records, r=register containing length of output record
	SEPASMB= YES	If DTF is assembled separately	
	SSELECT=n	For stacker selection to pocket other than NR or NP	n=stacker select character
	TYPEFLE= INPUT OUTPUT CMBND	If other than input	Specifies an input, output, or combined file
	WORKA=YES		If I/O records are processed in work areas, answer YES

Figure 18. DTFCD Entries

EOFADDR=Name

This entry specifies the symbolic name of the user's end-of-file routine. IOCS will automatically branch to this routine on an end-of-file condition. This entry must be specified for input and combined files. In his routine, the programmer can perform any operations required for the end of the file, and he generally issues the CLOSE instruction for the file.

IOCS detects end-of-file conditions in the card reader by recognizing /* punched in card columns 1 and 2 if the card reader is assigned to a programmer unit, or by recognizing either /* or /& if the unit is assigned to SYSRDR or SYSIPT. If cards are allowed to run out without a /* or /& trailer card, an error condition (intervention required) is signaled to the operator.

IOAREA1=Name

This entry specifies the symbolic name of the input or output area used by this file. An address expression, Name, is defined.

If the file is a combined file, this entry specifies the input area. If IOAREA2 is not specified, the area specified in this entry is used for both input and output.

IOAREA2=Name

This entry specifies a second I/O area. An address expression is defined. If the file is a combined file, this area will be the output area if it is specified.

IOREG=(r)

If work areas are not to be used and two input or output areas are used, specify the register (2-12) in which IOCS puts the address of the record. For output files, IOCS puts the address where the user can build a record. This entry may not be used for combined files.

MODNAME=Name

This entry may be used to specify the name of the logic module that will be used with the DTF table to process the file. If the logic module is assembled with the program, the MODNAME in the DTF macro instruction must specify the same name as the CDMOD macro instruction. If this entry is omitted, standard names will be generated for calling the logic module. If two DTF macro instructions call for different functions that can be handled by a single module, only this one module will be called

OUBLKSZ=n

This entry is used in conjunction with IOAREA2 only for a combined file. Enter the maximum number n of characters that will be transferred at one time. If this entry is not included and IOAREA2 is specified, the same length as BLKSIZE is assumed.

RECFORM={FIXUNB / UNDEF / VARUNB}

This entry specifies the record format of the file. If the record format is FIXUNB, this entry may be omitted.

If TYPEFLE=CMBND, this entry must be FIXUNB.

RECSIZE=(r)

For undefined records this entry specifies the register (2-12) that will contain the length of the output record. The user must load the length of each record into the register before he issues the PUT instruction for the record.

SEPASMB=YES

Specify this entry if the DTF is assembled separately. This causes a CATALR card with Filename to be punched ahead of the object deck and defines the Filename as an ENTRY point in this Assembly.

SSELECT=n

This entry specifies the stacker-select character that is valid for the file. If this entry is not specified, cards will be selected into NR (normal read) or NP (normal punch). This entry is not applicable to a combined file.

TYPEFLE={INPUT / OUTPUT / CMBND}

This entry specifies if the file is input, output, or combined. A combined file can be specified for an IBM 1442 or 2520 or for a 2540 with the punch-feed-read feature. TYPEFLE=CMBND is applicable if both GET's and PUT's are issued to the same card file.

WORKA=YES

If I/O records are to be processed in work areas instead of the I/O area, this entry is specified with YES. The programmer must set up the work area in main storage. The address expression of the work area or a general purpose register must be specified in each GET and PUT.

PARAMETERS AND NAMES FOR CDMOD (CARD MODULE)

Listed here are the user-supplied parameters for CDMOD. The first card contains CDMOD in the operation field and may contain a module name in the name field.

CONTROL=YES

Include this entry if the CNTRL macro instruction is to be used with the module and the DTF's associated with the module. The module will also process files in which the CNTRL macro is not used.

If CONTROL is specified, the CRLCHR parameter may not be specified. This parameter cannot be specified if IOAREA2 is used for an input file.

CRDERR=RETRY

Include this entry if the module is to include error retry routines for the 2540 and 2520 punch-equipment check. Whenever this parameter is specified, any DTF to be used with the module must also specify the CRDERR parameter.

This entry does not apply to an input or a combined file.

CTLCHR={YES / ASA}

Include this entry if first character stacker-select control will be used. Either YES or ASA may be specified. Whenever this parameter is included, any DTF to be used with the module must also specify the CTLCHR parameter with the appropriate YES or ASA operand.

If CTLCHR is included, CONTROL may not be specified.

This entry does not apply to a combined file.

DEVICE={2540 / 1442 / 2520 / 2501}

Include this entry to specify the I/O device that will be used by the module. Any DTF used with the module must have the same operand.

IOAREA2=YES

Include this entry if a second I/O area will be used. Any DTF to be used with the module must also include the IOAREA2 parameter. This entry is not required for combined files.

RECFORM={FIXUNB / VARUNB / UNDEF}

Specify the record format: fixed-length, variable-length, or undefined. Any DTF to be used with the module must include the appropriate operand in the RECFORM parameter.

For INPUT and COMBND files only, FIXUNB should be specified.

SEPASMB=YES

Include this parameter if the logic module is assembled separately. This causes a CATALR card with the module name (standard or user) to be punched ahead of the object deck.

TYPEFLE={INPUT / OUTPUT / CMBND}

This entry causes generation of a module for either input, output, or combined file. Any DTF to be used with the module must include the appropriate operand in the TYPEFLE parameter.

WORKA=YES

This entry is to be included if records are to be processed in work areas instead of I/O areas. Any DTF to be used with the module must include the appropriate operand in the WORKA parameter.

Recommended Module Name for CDMOD

Each name begins with a 3-character prefix (IJC) and consists of a 5-character field corresponding to the options permitted in the generation of the module.

CDMOD name = IJCabcde

a = F if RECFORM=FIXUNB (always for INPUT and CMBND files)
  = V if RECFORM=VARUNB
  = U if RECFORM=UNDEF

b = A if CTLCHR=ASA is specified (not specified if CMBND)
  = Y if CTLCHR=YES is specified
  = C if CONTROL=YES is specified
  = Z if neither CTLCHR nor CONTROL is specified

c = I if TYPEFLE=INPUT
  = O if TYPEFLE=OUTPUT
  = C if TYPEFLE=CMBND

d = Z if neither WORKA nor IOAREA2 is specified
  = W if WORKA=YES is specified
  = I if IOAREA2=YES
  = B if both WORKA and IOAREA2 are specified.

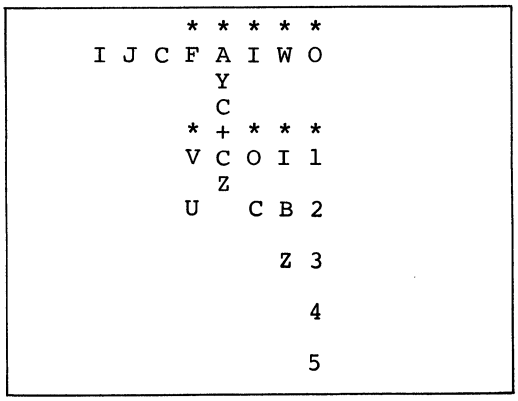
  For CMBND Files

    if WORKA=YES is specified, d = W
    if WORKA=YES is not specified, d = Z

e = 0 if DEVICE=2540
  = 1 if DEVICE=1442
  = 2 if DEVICE=2520
  = 3 if DEVICE=2501
  = 4 if DEVICE=2540 and CRDERR is specified.
  = 5 if DEVICE=2520 and CRDERR is specified.

Subsetting and Supersetting of CDMOD Names

The following diagram illustrates the subsetting and supersetting allowed for CDMOD names. All but one of the parameters are exclusive (do not allow supersetting). A module name specifying C (CONTROL) in the b location is a superset of a module name specifying Z (no control or CTLCHR). A module with the name IJCFCIW0 is a superset of a module with the name IJCFZIW0.



\+ Subsetting/supersetting permitted.

* No subsetting/supersetting permitted.

PAPER TAPE FILE (DTFPT)

A DTF entry is included for every paper tape input file that is to be processed in the program. The first entry must be the DTFPT header entry. Enter the symbolic name of the file, Filename, in the name field and DTFPT in the operation field. The detail entries follow the DTFPT header card in any order. Keyword entries are contained in the operand field. Figure 19 lists the entries.

BLKSIZE=n

This entry specifies the length of the input area. For undefined records, this area must be at least one position larger than the longest record, and must accomodate all shift and delete characters included in the record. For fixed-length records, the area must be exactly the size of the record. If shift or delete characters are present in the record (the SCAN entry is specified), BLKSIZE will indicate the number of characters required by the user after translation and compression. OVBLKSZ will contain the number of characters to be read in to produce the BLKSIZE number.

OPERATION	OPERAND	MUST BE INCLUDED	REMARKS
DTFPT		For each file	Header card. Specify 7-character or less symbolic filename in name field.
	BLKSIZE=n	For each file	UNDEF: at least one greater than the longest record FIXUNB: size of every record
	DEVADDR=SYSnnn	For each file	Symbolic unit
	EOFADDR=Name	For each file	Name of user's EOF routine
	ERROPT=IGNORE SKIP Name		Error option. If absent, IOCS will terminate the job
	FTRANS=Name	For shifted codes	Name of user's figure shift translate table
	IOAREA1=Name	For each file	Name of input area
	IOAREA2=Name	For two input areas	Name of second input area
	IOREG=(r)	For two input areas	Register (2-12) containing current input area address
	LTRANS=Name	For shifted codes	Name of user's letter shift translate table
	MODNAME=Name	For name other than standard	
	OVBLKSZ=n	Only when FIXUNB format is used with SCAN	Number of characters to be read in to produce number specified in BLKSIZE. If omitted, will be equal to BLKSIZE
	RECFORM=FIXUNB UNDEF	If UNDEF format	
	RECSIZE=(r)		Register (2-12) containing final length of input records
	SCAN=Name	When shift and/or delete characters are in code	Name of user's scan table for shift and delete characters
	SEPASMB=YES	If the DTF is assembled separately	
	TRANS=Name	When unshifted codes are to be translated	Name of user's translate table
	WLRERR=Name		Name of user's wrong length record routine. If omitted, error handled as in ERROPT; if both entries omitted, WLR indication will be ignored.

Figure 19. DTFPT Entries

DEVADDR=SYSnnn

This entry specifies the symbolic unit (SYSnnn) to be associated with this logical file. An actual channel and unit will be assigned to the unit by an ASSGN card in the Job Control statement. The ASSGN statement contains the same symbolic name as DEVADDR.

EOFADDR=Name

This entry specifies the symbolic name of the user's end-of-file routine. IOCS will automatically branch to this routine on an end-of-file condition, if the end-of-file switch is set ON.

In his routine, the programmer can perform any operations required for the end of file and issue the CLOSE instruction for the file.

ERROPT={IGNORE / SKIP / Name}

This entry is specified if the user does not want a job terminated when a read error cannot be corrected by the operator. If the ERROPT entry is omitted and a read error occurs, IOCS will terminate the job.

The entry IGNORE allows IOCS to handle the record as if no errors were detected. If the entry SKIP is specified, IOCS will skip the record in error and will cause the next record to be read in.

If neither IGNORE nor SKIP are specified, the symbolic name of the user's error routine should be specified. On an error condition, IOCS will read in the complete record, including the error character(s), and will then branch to the user's error routine. At the end of his error routine, the user must return to IOCS by branching to the address in Register 14, and the next record will be read in. The programmer must not issue any GET instructions for records in the error block. If any other IOCS macros are contained in the error routine, the contents of Register 14 must be saved and restored.

FTRANS=Name

This entry must be included for every file using a shifted code. It specifies the symbolic name of a figure shift table that must be provided by the user. This table must conform to the specifications of the machine instruction TRANSLATE. The entry TRANS must be omitted.

IOAREA1=Name

This entry specifies the input area. Enter an address expression (name) that specifies the input area.

IOAREA2=Name

This entry specifies a second input area. Enter an address expression (name) that specifies the input area. With two input areas specified, IOCS will overlap the filling of one input area with the processing of the record in the other area.

IOREG=(r)

This entry must be included if two input areas are used. It specifies the register into which IOCS will put the address of the record that is available for processing. Any register from 2 through 12 may be used.

LTRANS=Name

This entry must be included for every file using a shifted code. It specifies the symbolic name of a letter shift table which must be provided by the user. This table must conform to the specifications of the machine instruction TRANSLATE. The entry TRANS must be omitted.

MODNAME=Name

This entry may be used to specify the name of the logic module that will be used with the DTF table to process the file. If the logic module is assembled with the program, the MODNAME in the DTFPT macro instruction must specify the same name as the PTMOD macro instruction. If this entry is omitted, standard names will be generated for calling the logic module.

OVBLKSZ=n

This entry specifies the number of characters to be read in, before translation and compression, to produce the number of characters specified in the BLKSIZE entry. This entry is used only when SCAN and RECFORM=FIXUNB are both specified. If it is omitted, the number of characters to be read will be made equal to the number specified in the BLKSIZE entry.

RECFORM={FIXUNB / UNDEF}

This entry specifies the record format of this file. Either format may be specified for shifted and unshifted codes. If the record format is FIXUNB, this entry may be omitted.

RECSIZE=(r)

This entry specifies the number of the register that will contain the length of the input record. This may be any register from 2 through 12. This entry is optional. If it is present, IOCS will place the length of each record read into the register specified. If input files contain shift codes or other characters requiring deletion, the length placed in the register will be that of the compressed record.

SCAN=Name

This entry must be included for all files using shifted codes. It may also be included if the user wishes to delete certain characters from each record. The entry specifies the symbolic name of a table that must be provided by the user. This table, which must conform to the specifications of the machine instruction TRANSLATE AND TEST, will contain nonzero entries for all delete characters and, where appropriate, for the figure and letter shift characters. The entry in the table for the figure shift character must be hexadecimal 04, and for the letter shift character hexadecimal 08. Delete entries must be hexadecimal 0C. All other entries in the table must be zero. Any deviation from this will cause incorrect translation and may produce a program check. The table must be large enough to accomodate the maximum binary value of coding in the tape being processed; i.e., 255 bytes for eight-track tape. This prohibits erroneous coding in the tape from causing a SCAN function in a location beyond the limits of the SCAN table.

SEPASMB=YES

Include this parameter if the DTF is assembled separately. This causes a CATALR card with the filename to be punched ahead of the object deck and defines the Filename as an ENTRY point in the assembly.

TRANS=Name

This entry must be included if a non-shifted code is to be translated into internal System/360 code. The FTRANS and LTRANS entries must be omitted if this entry is present. If none of these entries is present, no translation will take place. The TRANS entry specifies the symbolic name of a table provided by the user. This table must conform to the specifications of the machine instruction TRANSLATE.

WLRERR=Name

When IOCS determines a wrong-length record is present, it will branch to the symbolic name specified in the WLRERR entry. If this entry is not included and the ERROPT entry is included, IOCS will consider the error uncorrectable and will use the ERROPT option specified. Absence of both ERROPT and WLRERR entries will cause the wrong-length record to be accepted as a normal record. Wrong-length checking will not be performed for fixed-length records because a fixed number of characters will be read in each time. Overlength undefined records will be detected when the incoming record fills the input area. The input area must, therefore, be at least one position larger than the longest record anticipated.

At the end of the user's WLRERR routine, the user must return to IOCS by branching to the address in Register 14. IOCS will cause the next record to be read in. If any other IOCS macros are included in the record-length error routines, the user must save and restore the contents of Register 14 in the error routine.

CHARACTERISTICS OF A PAPER TAPE FILE

Record Formats

Fixed unblocked records and undefined records are the only formats supported for the paper tape reader. Shifted and non-shifted codes are acceptable in both formats.

FIXED UNBLOCKED RECORDS: Fixed unblocked records should not have an End-of-Record (EOR) character following them. If they do, the EOR character will enter main storage as a normal character and will not stop

the data transfer. The user must define the number of characters contained in each record in the BLKSIZE entry. A count-controlled read will cause the number of characters specified (or fewer if an EOF condition occurs) to be read in as the result of a GET instruction. If shift codes and deletion characters are included in the record, the record will be translated and compressed. Additional reads will be performed automatically by IOCS until a translated record of the specified length has been obtained. Control is then returned to the user.

Note that when the programmer uses the fixed unblocked record format, he must not clear the input area beyond the length specified in his BLKSIZE entry. The remainder of the input area may now contain part of the next record, which has been read in and will remain there until the next GET is issued.

UNDEFINED RECORDS: Each undefined record must have an EOR character following it. The user must define an input area and a BLKSIZE entry that is at least one position larger than the longest record anticipated. A modified read (under control of count and the EOR character) will cause the number of characters specified, up to the next EOR character, to be read in. This record will be translated and, if SCAN was specified, compressed. There will be no re-reads. If the record fills the input area, it is assumed to be overlength, and IOCS will take the wrong-length record option specified by the user.

Note: All processing is carried out in the user's input area(s). Work areas are not utilized unless the user moves the record to the work area. The GET Filename,Work-area form of the GET macro instruction cannot be used for a paper tape file; only the GET Filename form is acceptable. If the user has two input areas, he must reference fields within his record by means of displacements relative to the general register specified by him in his IOREG entry or by means of a DSECT.

Code Translation

The TRANS entry is used for records containing non-shifted codes and translation is performed directly into internal System/360 code. If the input tape is punched in EBCDIC code, no translation is required and all translation entries may be omitted.

If the input tape contains shifted codes, the FTRANS, LTRANS, and SCAN entrie must be included. IOCS assumes that the first record read from the input tape star in figure shift. Therefore, if the first record starts with letter shift coding, th user must be sure that the first character of the first record is a letter shift character. The shift status is carried from one record to the next and remains unchanged until another shift character is encountered.

Translation of shifted codes is accomplished as follows.

1. The record is first scanned for shift characters and the segment between the shift characters is translated using the appropriate shift table.

2. The translated segment is moved to the left to remove the shift character.

3. The above steps are repeated for each segment until the complete record has been translated and compressed.

These steps result in a translated and compressed record left justified in the in put area. The record length is communicat to the user in the register designated in the RECSIZE entry, if present.

If the record format is fixed unblocked the number of characters specified in the OVBLKSZ entry will be read in, translated, and compressed. If the resulting record i shorter than that specified in the BLKSIZE entry, additional reads will be performed until the record length is equal to, or greater than, the BLKSIZE specification. The record is then available to the user. If the final read results in a record leng that exceeds the BLKSIZE specification, th remaining characters are moved into the be ginning of the input area, as the start of the next record, when the next GET is issu

SCAN may be used alone, or in conjuncti with TRANS, to delete characters from records that do not contain shifted code. In this case, there must be no 04 or 08 entries in the SCAN table.

The EOR character must be independent o shift status. That is, it must be effecti whether the coding is in letter or figure shift. If there is valid character coding in either shift which corresponds to the EOR coding established for a particular job, the corresponding code must not be in cluded in the input record.

rror and Wrong-Length Record Conditions

he paper tape reader stops immediately on n error condition. If the error cannot be orrected and the job is not to be termiated, IOCS will cause the complete record ontaining the error to be translated and ompressed before taking the error option pecified by the user.

The only wrong-length record condition hat can be detected is an overlength unefined record when at least one additional osition has been provided in the input rea and this fact has been reflected in he BLKSIZE entry. It is the operator's esponsibility to determine that the input ape is correctly positioned for the next ET instruction, at the physical beginning f the next record.

Wrong-length record indication is not ossible with fixed unblocked records since ach record is a sequence of a specified umber of characters. The FIXUNB record ormat should be used with caution since ne character too few or too many in any ecord will cause all subsequent records o be out of phase.

The last record on any file should be hecked for correct length by the user hrough the use of the RECSIZE entry. A ecord cannot be partly on one reel of inut tape and partly on another reel.

rogramming Considerations

hen processing fixed unblocked records ith logical IOCS using the SCAN option, ata checks must not be ignored. Logical OCS prohibits the operator from ignoring uch an error because ignoring the error ould result in processing all subsequent ecords out of phase (on other than the inended record boundaries). LIOCS cannot etermine whether the error character was shift or delete character, or whether it hould be included in the record length ount.

When processing any other type of record ith LIOCS and a data check occurs, the rror can be ignored if the programmer so irects the operator.

To allow the possibility of ignoring lata checks when processing with physical OCS (using EXCP and WAIT instructions), he programmer must set byte 2, bit 4 of he CCB ON.

Note: Immediately after using the WAIT instruction, the programmer must restore the CCW.

PARAMETERS AND NAMES FOR PTMOD (PAPER TAPE MODULE)

Listed here are the user-supplied parameters for PTMOD. The first card contains PTMOD in the operation field and may contain a user module name in the name field.

RECFORM={FIXUNB / UNDEF} Required only if the entry SCAN=YES is present. If records of undefined format using the SCAN option are to be translated, the UNDEF parameter must be specified. If records are of fixed unblocked format, the FIXUNB parameter may be specified or it may be omitted.

SCAN=YES Required for records containing shift characters or characters which are to be automatically deleted.

SEPASMB=YES Include this parameter if the logic module is assembled separately. This causes a CATALR card with the module name (standard or user) to be punched ahead of the object deck.

TRANS=YES Required only if the entry SCAN=YES is not present. If it is specified, a module will be generated which will translate records using an unshifted code.

Summary of PTMOD

The following are the only possible combinations of entries.

1. No parameters specified. Module does not provide routines for translation nor for shift or delete characters.
2. TRANS=YES. Module handles translation of unshifted code, but no delete characters.

3. SCAN=YES [RECFORM=FIXUNB]. Module handles shift and delete characters for records of fixed unblocked format.

4. SCAN=YES, RECFORM=UNDEF. Module handles shift and delete characters for records of undefined format.

Recommended Module Name for PTMOD

Each name begins with a 3-character prefix (IJE) and consists of a 5-character field corresponding to the options permitted in the generation of the module.

PTMOD name = IJEabcde

a = S if SCAN=YES is specified
  = Z if SCAN=YES is not specified

b = T if TRANS=YES is specified, and SCAN=YES is not specified
  = Z if TRANS=YES is not specified

c = F if RECFORM=FIXUNB, and SCAN=YES is specified
  = U if RECFORM=UNDEF, and SCAN=YES is specified
  = Z if SCAN=YES is not specified

d = Z always

e = Z always

Subsetting and Supersetting of PTMOD Names

The following diagram illustrates the PTMOD names. No subsetting or supersetting is allowed.

```
                * * *
          I J E Z Z Z Z Z

                Z T Z

                S Z F

                S Z U

* No subsetting/supersetting permitted
```

PRINTER FILE (DTFPR)

A DTFPR entry is included for each printer file that is processed in the program. The first entry is the DTFPR header entry. The name field contains the symbolic file name, Filename. The operation field contains DTFPR. The detail entries, in any order, follow the DTFPR header entry with keyword entries in the operand field. Figure 20 contains DTFPR entries.

BLKSIZE=n

This entry specifies the length of IOAREA1. If the record format is variable or undefined, enter the length of the longest record. If this entry is omitted, 121 is assumed.

CONTROL=YES

This entry is specified if the CNTRL macro will be issued to the file. If this parameter is specified, CTLCHR must be omitted.

CTLCHR={YES / ASA}

This entry is specified if first-character control is to be used. The entry CTLCHR=ASA specifies the American Standard Association set. The entry CTLCHR=YES specifies the System/360 character set. Appendix B contains the codes. If this parameter is specified, CONTROL must be omitted.

DEVADDR=SYSnnn

This entry specifies the symbolic unit (SYSnnn) to be associated with this printer.

DEVICE=

This entry specifies that one of the following printers is used for the file: 1403, 1404 (continuous forms only), 1443 or 1445. Enter one of these numbers. If this entry is omitted, 1403 will be the assumed device.

IOAREA1=Name

This entry specifies the output area. An address expression (Name) is specified.

IOAREA2=Name

This entry specifies a second output area. An address expression (Name) is specified.

OPERATION	OPERAND	MUST BE INCLUDED	REMARKS
DTFPR		For each file	Header card. Specify 7-character or less symbolic filename in name field
	BLKSIZE=n	If other than 121	n = length of I/O area. If the record is not fixed, enter the length of the longest record.
	CONTROL YES	If a CNTRL macro will be used	
	CTLCHR= YES ASA	For first character forms control.	YES=S/360 character set ASA=American Standard Association set
	DEVADDR= SYSnnn	For each file	Symbolic unit
	DEVICE= 1403 1404 1443 1445	For file other than 1403.	Actual device
	IOAREA1 name	For each file	Specifies name of output area
	IOAREA2= name	If two I/O areas are specified	Specifies name of second output area
	IOREG=(r)	For two output areas	r=register 2-12
	MODNAME= name	If name other than standard or a more inclusive module is referenced	Specifies a user-named I/O module
	PRINTOV= YES	If PRTOV macro is used	
	RECFORM= FIXUNB UNDEF VARUNB	If record is not FIXUNB	
	RECSIZE=(r)	For undefined records	r=register 2-12 containing length of output record
	SEPASMB= YES	If DTF is assembled separately	
	UCS = $\frac{\text{OFF}}{\text{ON}}$		OFF allows checks ON disallows data checks
	WORKA= YES	If records are processed in work areas	

Figure 20. DTFPR Entries

IOREG=(r)

If two output areas and no work areas are used, the entry IOREG=(r) specifies the address of the area where the user can build a record. The (r) represents a register 2-12.

MODNAME=Name

This entry may be used to specify the name of the logic module that will be used with the DTF table to process the file. If the logic module is assembled with the program, the MODNAME in the DTF macro instruction must specify the same name as the PRMOD macro instruction. If this entry is omitted, standard names will be generated for calling the logic module. If two DTF macro instructions call for different functions that can be handled by a single module, only one module will be called.

PRINTOV=YES

This entry is specified if the PRTOV macro instruction is included in the problem program.

RECFORM={FIXUNB / UNDEF / VARUNB}

The entry RECFORM=FIXUNB is specified if the record format is fixed. When the record format is FIXUNB, this entry may be omitted. The entry RECFORM=UNDEF is specified if the record format is undefined. If the output is variable and unblocked, enter VARUNB.

RECSIZE=(r)

This entry, for undefined records, specifies the general register (2-12) that will contain the length of the output record. The user must load the length of each record into the register before he issues the PUT instruction.

SEPASMB=YES

Include this parameter if the DTF is assembled separately. This causes a CATALR card with the Filename to be punched ahead of the object deck and defines the Filename as an ENTRY point in the assembly.

UCS={ON / OFF}

This entry determines whether data checks are to be allowed or disallowed on a 1403 printer with the Universal Character Set feature. The entry is especially useful to programmers who are using first-character forms control and who have modules that cannot process the CNTRL macro.

OFF	Allow data checks.
ON	Disallow data checks.

WORKA=YES

If output records are processed in work areas instead of the output area, the entry WORKA=YES is specified. The user must set up the work area(s) in main storage. The address expression of the work area (or a general register containing the address) must be specified for each GET or PUT.

PARAMETERS AND NAMES FOR PRMOD (PRINTER MODULE)

Listed here are the user-supplied parameters for PRMOD. The first card contains PRMOD in the operation field and may contai a user module name in the name field.

CONTROL=YES

Include this entry if CNTRL macro instructions are to be used with the associated DTF's. The module will also process files which do not use the CNTRL macro instruction.

If CONTROL is specified, the CTLCHR parameter should not be specified.

CTLCHR={YES / ASA}

Include this entry if first-character carriage control is to be used. Whenever this parameter is included, any DTF to be used with the module must also specify the CTLCHR parameter with the appropriate YES or ASA operand.

If CTLCHR is included CONTROL should not be specified.

IOAREA2=YES — Include this entry if a second I/O area is to be used. Any DTF to be used with the module must also include the IOAREA2 parameter.

?RINTOV=YES — Include this entry if PRTOV macro instructions are to be used with the associated DTF's. The module will also process any files which do not use the PRTOV macro instruction.

RECFORM={FIXUNB / VARUNB / UNDEF} — This entry causes generation of a module which will process the specified record format: fixed-length, variable-length, or undefined. Any DTF to be used with the module must include the appropriate operand in the RECFORM parameter.

SEPASMB=YES — Include this parameter if the logic module is assembled separately. This causes a CATALR card with the module name (standard or user) to be punched ahead of the object deck.

WORKA=YES — Include this entry if records are to be processed in work areas instead of I/O areas. Any DTF to be used with the module must include the appropriate operand in the WORKA parameter.

Recommended Module Name for PRMOD

Each name begins with a 3-character prefix (IJD) followed by a 5-character field corresponding to the options permitted in the generation of the module.

PRMOD name = IJDabcde

a = F if RECFORM=FIXUNB
  = V if RECFORM=VARUNB
  = U if RECFORM=UNDEF

b = A if CTLCHR=ASA is specified
  = Y if CTLCHR=YES is specified
  = C if CONTROL=YES is specified
  = Z if neither CTLCHR nor CONTROL is specified

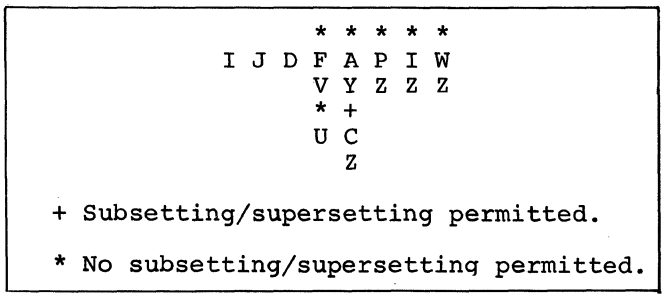
c = P if PRINTOV=YES is specified
  = Z if PRINTOV=YES is not specified

d = I if IOAREA2=YES is specified
  = Z if IOAREA2=YES is not specified

e = W if WORKA=YES is specified
  = Z if WORKA=YES is not specified

Subsetting and Supersetting of PRMOD Names

The following diagram illustrates the subsetting and supersetting allowed for PRMOD names. Two of the parameters allow subsetting. For example, the module name IJDFCPIW is a superset of the module names IJDFCZIW and IJDFZZIW.



MAGNETIC TAPE FILES (DTFMT)

A DTFMT entry is included for each magnetic tape input or output file that is to be processed. The DTFMT header entry is followed by a series of detail entries that describe the file (Figure 21). The detail entries generate the DTF table. Enter the symbolic name of the file in the name field and DTFMT in the operation field. The entries following the header entry may appear in any order.

BLKSIZE=n

Enter the length n of the I/O area. If the record format is variable or undefined, enter the length of the largest block of records. If a READ or WRITE macro specifies a length greater than n for workfiles, the record length will be greater than BLKSIZE. The maximum block size is 32,767 bytes (32K minus one). The minimum size physical tape record (gap to gap) is 12 bytes. Eleven bytes or less are considered a noise record.

CKPTREC=YES

This entry is necessary if a tape input file will have checkpoint records interspersed among the data records. IOCS will bypass any checkpoint records that are encountered.

OPERATION	OPERAND	MUST BE INCLUDED	REMARKS
DTFMT		For each file	Header card. Specify 7-character or less symbolic filename in name field.
	*BLKSIZE=n	For each file	n=length of the I/O area. If the record is not fixed enter the length of the longest record.
	CKPTREC= YES	If input file has checkpoint records among data records	IOCS will bypass checkpoint records.
	*DEVADDR= SYSnnn	For each file	Specifies symbolic unit
	*EOFADDR= name	Input or work files	Specifies name of user's end-of-file routine.
	*ERROPT= IGNORE SKIP name	For error correction	IGNORE allows IOCS to handle the record as if no errors were detected. SKIP allows IOCS to bypass the record. Name is the name of a routine to which the program will branch.
	FILABL= STD NSTD NO	For each file	Specifies the type of label processing desired.
	IOAREA1= name	For each file	Specifies I/O area. Name=address expression.
	IOAREA2= name	If two I/O areas are specified	Specifies a second I/O area. Name=address expression
	IOREG=(r)		r=register (2-12) if records are blocked or processed in the I/O area.
	LABADDR= name	To process user's labels	name=symbolic name of user's routine
	MODNAME =name	If name other than standard or a more inclusive module is referenced	Specifies a user named I/O module
	*NOTEPNT= YES POINTS	YES is required if NOTE, POINTR, POINTW, or POINTS is used	
	*READ= FORWARD BACK	If READ=BACK is specified	
	*RECFORM= * FIXUNB FIXBLK VARUNB VARBLK * UNDEF	For other than FIXUNB	

Figure 21. DTFMT Entries (Part 1 of 2)

OPERATION	OPERAND	MUST BE INCLUDED	REMARKS
	RECSIZE=n or (r)	For FIXBLK or UNDEF	n=number of characters in each record for fixed-length blocked records. r=register (2-12) containing record length for undefined records.
	*REWIND= UNLOAD NORWD		Specifies procedure for rewinding tape.
	SEPASMB= YES	If DTF is assembled separately	
	*TYPEFLE= INPUT OUTPUT *WORK	For output or work files	Specifies whether file is input or output or if a work file is specified.
	TPMARK= NO	If a tapemark is not to be the first record of a tape output file when no labels are specified, or if no tapemark is to be written following nonstandard labels.	
	VARBLD= (r)	For variable-length blocked records.	r=general register (2-12). If WORKA=YES is specified VARBLD will be ignored.
	WLRERR= name	To branch to user's routine to handle wrong-length error correction.	name=symbolic name of user's routine. If omitted, error handled as in ERROPT; or if both entries omitted, WLR indication will be ignored.
	WORKA= YES	If records are processed in work areas	

* Entries for work files

Figure 21. DTFMT Entries (Part 2 of 2)

DEVADDR=SYSnnn

This entry specifies the symbolic unit (SYSnnn) to be associated with the logical file. An actual channel and unit number will be assigned to the unit by an ASSGN statement. The ASSGN card contains the same symbolic name as DEVADDR.

EOFADDR=Name

This entry specifies the symbolic name of the user's end-of-file routine. IOCS will automatically branch to this routine on an end-of-file condition. This entry must be specified for input and work files.

In his routine, the programmer can perform any operations required for the end of file, and he generally issues the CLOSE instruction for the file.

IOCS detects end-of-file conditions in magnetic tape input by reading a tapemark and EOF when standard labels are specified or /* if the unit is assigned to SYSRDR or SYSIPT. If standard labels are not specified, IOCS assumes an end-of-file condition when the tapemark is read. The user must determine, in his routine, that this actually is the end of the file.

ERROPT={IGNORE / SKIP / Name}

This entry specifies functions to be performed for an error block.

If a parity error is detected when a block of tape records is read, the tape is backspaced and reread 100 times before the tape block is considered an error block.

If either FILABL=STD or CKPTREC, or both, is specified, the error block is included in the block count that is taken. After this the job is automatically terminated, unless this ERROPT entry is included to specify other procedures to be followed on an error condition. Either IGNORE, SKIP, or the symbolic name of an error routine can be specified in this card. The functions of these three specifications are:

IGNORE — The error condition is completely ignored, and the records are made available to the user for processing.

SKIP — No records in the error block are made available for processing. The next block is read from tape, and processing continues with the first record of that block. The error block is included in the block count.

Name — IOCS branches to the user's routine, where he may perform whatever functions he desires to process or make note of the error condition. Register 1 contains the address of the block in error, and Register 14 contains the return address.

In his routine, the programmer should address the error block, or records in the error block, by referring to the address supplied in Register 1. The contents of the IOREG register or the work area (if either is specified) may vary and therefore should not be used for error blocks. Also, the programmer must not issue any GET instructions for records in the error block. If he uses any other IOCS macros in his routine, he must save and restore the contents of registers 0, 1, 14, and 15. At the end of his error routine, he must return to IOCS by branching to the address in register 14. When control is returned to the problem program, the first record of the next block is available for processing in the main program.

This ERROPT entry does not apply to tape output files. The job is automatically terminated if a parity error still exists after IOCS attempts 15 times to write a tape output block. This includes erasing forward.

This entry applies to wrong-length records if the entry WLRERR is not included. If both ERROPT and WLRERR are omitted and wrong length records occur, IOCS will assume the IGNORE option.

FILABL={STD / NO / NSTD}

The entry STD is used to indicate that standard labels will be processed. Enter NO if no labels are contained on the file.

If nonstandard labels are contained on the file, enter NSTD. The user must furnish a routine to check or create the nonstandard labels by using his own I/O area and EXCP to read or write the labels. The entry point of this routine is the operand of LABADDR.

IOAREA1=Name

This entry specifies the I/O area. Enter an address expression (name) which specifies the I/O area. When variable-length records are processed, the size of the I/O area must include four bytes for the block size.

IOAREA2=Name

This entry specifies a second I/O area. Enter an address expression (name) which specifies the I/O area. When variable-length records are processed, the size of the I/O area must include four bytes for the blocksize.

IOREG=(r)

If two input or output areas are used, if blocked input or output records are processed in the I/O area, if variable unblocked records are read, or if undefined records are read backwards, this entry specifies the register in which IOCS places the address of the record that is available for processing. For output files, IOCS places, in a register, the address of the area where the user can build a record. An register (2-12) may be specified.

LABADDR=Name

Enter the symbolic name of the user routine to process user-standard or nonstandard labels. See the discussion under Tape OPEN Writing and Checking User Standard Labels and Writing and Checking Nonstandard Labels

MODNAME=Name

This entry specifies the name of the logic module that will be used with the DTF table to process the file. If the logic module was assembled with the program, the MODNAME in the DTF macro instruction must specify the same name as the MTMOD macro instruction. If this entry is omitted, standard names will be generated for calling the logic module. If two DTF macro instructions call for different functions that can be handled by a single module, only one module will be called. For example, if one DTF specifies READ=FORWARD and another specifies READ=BACK, only one logic module capable of handling both functions will be called.

NOTEPNT={POINTS / YES}

The entry YES is specified if the NOTE, POINTW, POINTR, or POINTS macro instructions will be issued to the tape work file. If POINTS is specified, only POINTS macro instructions can be issued to tape work files.

READ={FORWARD / BACK}

This entry specifies the direction in which the tape is read. READ=FORWARD may be omitted.

RECFORM=

This entry specifies the type of records (fixed- or variable-length, blocked or unblocked, or undefined) in the input or output file. One of the following may be entered immediately following the = sign.

FIXUNB for fixed-length unblocked records

FIXBLK for fixed-length blocked records

VARUNB for variable-length unblocked records

VARBLK for variable-length blocked records

UNDEF for undefined records.

If the record format is fixed-length unblocked, this entry may be omitted. Work files may use only FIXUNB or UNDEF.

RECSIZE=n or (r)

For fixed-length blocked records, this entry is required. It specifies the number of characters, n, in each record.

For undefined records, this entry is required for output files and optional for input files. It specifies a general register (2-12) that contains the length of the record. On output, the user must load the length of each record into the register before he issues the PUT. If specified for input, IOCS will provide the length of the record transferred to main storage.

REWIND={UNLOAD / NORWD}

If this specification is not included, tape files are automatically rewound, but not unloaded, on an OPEN or CLOSE instruction or on an end-of-volume condition. If other operations are desired for a tape input or output file, this entry specifies:

UNLOAD to rewind the tape on OPEN or to rewind and unload on a CLOSE or end-of-volume condition.

NORWD to prevent rewinding the tape at any time.

SEPASMB=YES

Include this parameter if the DTF is assembled separately. This causes a CATALR card with the Filename to be punched ahead of the object deck and defines the Filename as an ENTRY point in the assembly.

TPMARK=NO

This entry is included if the user does not want a tape mark written as the first record on a tape output file when no labels are specified. This entry is also included if no tape mark is to be written following nonstandard labels. If this entry is omitted for a tape output file, a tape mark will be the first record if no labels are specified. Also if this entry is omitted a tape mark will be written following nonstandard labels.

TYPEFLE={INPUT / OUTPUT / WORK}

Use this entry to indicate whether the file is an input or output file. If INPUT is specified, the GET macro will be used. If OUTPUT is specified, the PUT macro will be used. If WORK is specified, the READ/WRITE, NOTE/POINT, and CHECK macros will be used. (See the section Work Files for DTFMT and DTFSD.)

VARBLD=(r)

Whenever variable-length blocked records are built directly in the output area (no work area specified), this entry must be included. It specifies the number (r) of a general-purpose register, which will always contain the length of the available space remaining in the output area. Any register (2-12) may be specified.

Only after the PUT instruction is issued for a variable-length record, IOCS calculates the space still available in the output area and supplies it to the programmer in the VARBLD register. The programmer then compares the length of his next variable-length record with the available space to determine if the record will fit in the area. This check must be made before the record is built. If the record will not fit, the programmer issues a TRUNC instruction to transfer the completed block of records to the tape file. Then the present record is built at the beginning of the output area, as the first record of the next block.

WLRERR=Name

This entry applies only to tape input files. It specifies the symbolic name of a user's routine to receive control if a wrong-length record is read.

The address of the record is supplied by IOCS in register 1. In his routine the user can perform any operation he desires for wrong-length records. However, he must not issue any GET macro instructions. If he uses any other IOCS macros in his routine, he must save and restore the contents of register 14. At the end of his routine the user must return to IOCS by branching to the address in register 14.

When fixed-length unblocked records are specified (RECFORM=FIXUNB), a wrong length record error condition is given when the length of the record read is not equal to that specified in the BLKSIZE parameter. For fixed-length blocked records, record length is considered incorrect if the physical tape record (gap to gap) that is read is not a multiple of the logical-record length (specified in DTF RECSIZE), up to the maximum length of the block (specified in DTFMT BLKSIZE). This permits the reading of short blocks of logical records without a wrong-length-record indication.

For variable-length records blocked and unblocked, record length is considered incorrect if the length of the tape record is not the same as the block length specified in the four byte block-length field.

The WLRERR option is taken for undefined records if the record read is greater than the size specified by the BLKSIZE parameter.

If the WLRERR entry is omitted but a wrong-length record is detected by IOCS, one of the following will result.

- If the ERROPT entry is included for this file, the wrong-length record will be treated as an error block and handled according to the user's specifications for an error (IGNORE, SKIP, or Name-of-error routine).
- If the ERROPT entry is not included, IOCS will assume the IGNORE option of ERROPT.

WORKA=YES

If I/O records are processed in work areas instead of the I/O area, specify YES with this entry. The user must set up the work area in main storage. The address expression of the work area, or general register containing the address, must be specified in each GET or PUT.

PARAMETERS AND NAMES FOR MTMOD (MAGNETIC TAPE MODULE)

Listed here are the user-supplied parameters for MTMOD. The first card contains MTMOD in the operation field and may contain a user module name in the name field.

CKPTREC=YES

Include this entry if tape input files to be processed by the module will contain checkpoint records interspersed among the data records. The module will also process files which do not have checkpoint records, i.e., those whose DTF's do not specify CKPTREC=YES.

This entry is not needed for work files.

ERROPT=YES

Include this entry if the module is to handle any of the error options for an error block. Logic is generated to handle any of the three options (IGNORE, SKIP, or name). The module will process any files in which the ERROPT parameter is not specified in the DTF.

This entry is needed for work files, but it is not needed for input or output files.

NOTEPNT={YES / POINTS}

Include this entry if NOTE/POINT logic will be used with the module. If YES, the module will process any NOTE, POINTR, POINTW, or POINTS macro instruction. If POINTS is specified, only the POINTS macro instruction will be processed.

Modules specifying either one of the two options will also process work files for which the NOTE/POINT parameter is not specified. Modules specifying YES will also process work files specifying only POINTS. This entry does not apply to input or output files.

READ={FORWARD / BACK}

This entry causes generation of a module that will read tape files forward or backward.

If the operand specifies FORWARD, logic to read tape forward only is generated. Any DTF used with the module may not specify BACK in the READ parameter statement.

If the operand is BACK, logic to read tape both forward and backward is generated, and any DTF used with the module may specify either FORWARD or BACK as its READ parameter.

This entry is not needed for work files.

RECFORM={FIXUNB* / FIXBLK* / VARUNB / VARBLK / UNDEF}

This entry causes generation of an input/output module which will process either fixed-length, variable-length or undefined records. If either FIXUNB or FIXBLK is specified, a logic module will be generated which allows processing of both fixed-length record types. If either VARUNB or VARBLK is specified, a logic module will be generated which allows processing of both variable-length record types. If UNDEF is specified, a logic module for processing undefined record types will be generated. Any DTF used with the module must specify the same record format type as the module. For example, if the module has the entry RECFORM=FIXUNB, the DTF may have either the entry RECFORM=FIXUNB or RECFORM=FIXBLK.

This entry is not needed for work files.

* FIXBLK and FIXUNB use identical table formats and logic modules.

SEPASMB=YES

Include this parameter if the logic module is assembled separately. This causes a CATALR card with the module name (standard or user) to be punched ahead of the object deck.

TYPEFLE={INPUT / OUTPUT / WORK}

This entry causes generation of a logic module that will process either GET/PUT macro instructions or READ/WRITE, NOTE/POINT and CHECK macro instructions for work files. If the operand of the entry specifies WORK, then logic to process work files is generated; otherwise, a module to handle both input and output file types is assumed. Only DTF's for work files may be used with work file modules, and only DTF's for input

or output files may be used with an input/output module.

Note: INPUT and OUTPUT have the same table format and logic modules.

WORKA=YES — This entry is to be included if records are to be processed in work areas instead of I/O areas for the GET/PUT macros. The module will also process files which do not use a work area.

This entry is not needed for work files.

Recommended Module Name for MTMOD

Each name begins with a 3-character prefix (IJF) and consists of a 5-character field corresponding to the options permitted in the generation of the module.

In MTMOD there are two module classes: the module class for handling GET/PUT functions and the module class for handling READ/WRITE, NOTE/POINT, and CHECK functions (work files).

Name list for GET/PUT type modules.

MTMOD name = IJFabcde

a = F if RECFORM=FIXUNB (or FIXBLK)
  = V if RECFORM=VARUNB (or VARBLK)
  = U if RECFORM=UNDEF

b = B if READ=BACK is specified
  = Z if READ=FORWARD, or if READ is not specified
c = C if CKPTREC=YES is specified
  = Z if CKPTREC=YES is not specified

d = W if WORKA=YES is specified
  = Z if WORKA=YES is not specified

e = Z always

Name list for Work file type modules (TYPEFLE=WORK).

MTMOD name = IJFWabcd

a = E if ERROPT=YES is specified
  = Z if ERROPT is not specified

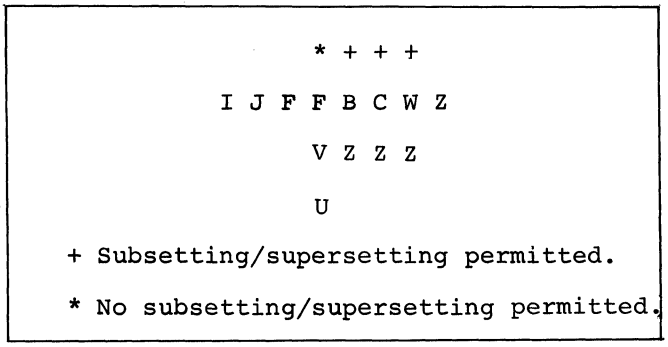
b = N if NOTEPNT=YES is specified
  = Z if NOTEPNT=YES is not specified
  = S if NOTEPNT=POINTS is specified

c = Z always

d = Z always

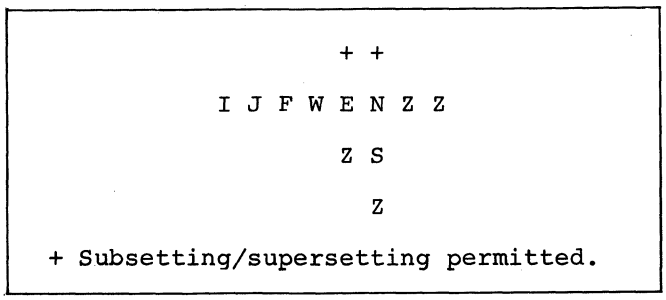
Subsetting and Supersetting of MTMOD Names

The following diagram illustrates the subsetting and supersetting allowed for MTMOD names. Three of the parameters allow subsetting. For example, the module name IJFFBCWZ is a superset of any MTMOD name specifying fixed-length records.



Subsetting and Supersetting Work Files for MTMOD

The following diagram illustrates the subsetting/supersetting relationship for work files using magnetic tape.



SEQUENTIAL DASD FILES (DTFSD)

The DTFSD macro instruction defines sequential (consecutive) processing for a file contained in a DASD. Only IBM standard label formats will be processed. The DTFSD macro instruction can be used with the IBM 2311 Disk Storage Drive or the IBM 2321 Data Cell Drive Model 1.

A DTFSD entry is included for each sequential input or output DASD file that is processed in the program (Figure 22). The DTFSD header entry and a series of detail entries describe the file. Symbolic addresses of routines and areas are specified in the detail entries.

Enter the symbolic name of the file in the name field and DTFSD in the operation field.

OPERATION	OPERAND	MUST BE INCLUDED	REMARKS
DTFSD		For each file	Header card, specify 7-character or less symbolic filename in name field.
	*BLKSIZE=n	For each file	Length of I/O area. n=maximum number of characters.
	*CONTROL= YES	CNTRL macro used	
	*DELETFL= NO	If CLOSE macro is not to delete the Format 1 and Format 3 label for a work file.	Applies to work files.
	*DEVICE= 2311 2321	For 2321	
	*EOFADDR= name	For input or work files	Symbolic name of end-of-file routine.
	*ERROPT= IGNORE SKIP name	For error correction on input or output files	Provides facility to handle errors. SKIP not allowed for output.
	IOAREA1= name	For each file	Symbolic name of I/O area.
	IOAREA2= name	If two I/O areas are used by GET and PUT.	Symbolic name of I/O area.
	IOREG=(r)	For blocked records processed in the I/O area, or if two I/O areas are used and the records are processed in the I/O areas.	Register 2-12 contains address of the record. r=register 2-12
	LABADDR= name	If user processes his own labels.	name=symbolic name of user's routine.
	MODNAME= name	If name other than standard or a more inclusive module is referenced.	Specifies a user-named I/O module.
	*NOTEPNT= YES POINTRW	If NOTE, POINTR or POINTW macros will be issued, POINTRW is specified. Specify YES for NOTE, POINTR, POINTW, and POINTS.	
	*RECFORM= * FIXUNB FIXBLK VARUNB VARBLK * UNDEF	For other than FIXUNB	Work files permit fixed unblocked or undefined records only.
	RECSIZE=n or (r)	Number of characters in record (fixed-length and blocked) or register 2-12 for undefined records	Required for each file.

Figure 22. DTFSD Entries (Part 1 of 2)

OPERATION	OPERAND	MUST BE INCLUDED	REMARKS
	SEPASMB= YES	If DTF is assembled separately	
	TRUNCS= YES	With TRUNC macro for Output FIXBLK file, or if FIXBLK DASD files contain short blocks within an input file.	
	*TYPEFLE= INPUT OUTPUT * WORK	For OUTPUT or WORK	
	UPDATE= YES	If DASD input file is updated	
	* VARBLD=(r)	Variable-length blocked records are built in output area	r=register 2-12
	*VERIFY= YES	To check a record written on 2311	Assumed for 2321
	WLRERR= name	To process wrong length records on input file	name=symbolic name of user's routine
	WORKA= YES	If records are to be processed in a work area	Omit this entry if IOREG (r)

*** Entries for a disk workfile.**

Figure 22. DTFSD Entries (Part 2 of 2)

The detail entries can follow in any order. Keyword entries are contained in the operand field.

BLKSIZE=n

Enter the length (n) of the I/O area. If the record format is variable or undefined, enter the length of the I/O area needed for the largest block of records.

CONTROL=YES

This entry is specified if a CNTRL macro is to be issued to the file. A CCW will be generated for control commands.

DELETFL=NO

Specify this parameter if the CLOSE macro is not to delete the Format-1 and Format-3 label for a work file. The parameter applies to work files only.

DEVICE={2311 / 2321}

This entry is included to specify whether the data file is located on an IBM 2311 or 2321. If unspecified, 2311 is assumed.

EOFADDR=Name

This entry specifies the symbolic name of the user's end-of-file routine. IOCS will automatically branch to this routine on an end-of-file condition. In his routine the user can perform any operations required for the end of the file. He generally issues the CLOSE there.

ERROPT={IGNORE / SKIP / Name}

This entry is specified if the user does not want a job terminated when a read or write error cannot be corrected in the disk error routines. If a parity error is detected when a block of records is read, the disk block is reread 10 times before it is considered an error block. After unsuccessfully reading 10 times, the job is terminated unless the ERROPT entry is included. Enter one of the following parameters after the = sign if the ERROPT entry is desired.

IGNORE	The error condition is ignored. The records are made available to the user for processing.
SKIP	No records in the error block are made available

for processing. The next block is read from the disk, and processing continues with the first record of that block.

Name

This entry specifies the name of the routine available to process the error condition. Register 1 contains the address of the error block. Register 14 contains the return address. The error block should be referenced using register 1. The contents of IOREG or the work area (if either is specified) may vary. Consequently, they should not be used to reference the error block. GET macro instructions must not be issued to the error block. If any other IOCS macros are used in the error routine, the contents of register 14 must be saved. To return to IOCS, the error routine branches to the address contained in register 14. After control is returned to the problem program, the first record of the next block is available for processing.

For an output file, IOCS branches to the specified error routine. Register 1 contains the address of the error block. Register 14 contains the return address of the problem program. The address is the instruction following the PUT (CHECK for work files) that detected the error.

On an output file, the only acceptable entries are IGNORE or Name. On an UPDATE=YES file, the entry SKIP will cause write errors to be ignored. The ERROPT entry applies to wrong-length records if the DTFSD entry WLRERR is not included.

IOAREA1=Name

This entry specifies the symbolic name of the I/O area to be used by the file. IOCS will read records into this area, or IOCS will write records from this area. When variable-length records are processed, the size of the I/O area must include four bytes for the block size. The I/O area must include eight bytes to build a count field for output files. If the file is an update file, the input and output areas are the same. If the record format for the file is variable length, the I/O area must begin on a half-word boundary.

IOAREA2=Name

If two I/O areas are to be used by GET or PUT, this entry is specified. Enter the symbolic name of the second I/O area to be used. When variable-length records are processed, the size of the I/O area must include four bytes for the block size. The I/O area must include eight bytes to build a count field for output files.

IOREG=(r)

This entry specifies the general purpose register (2-12) in which IOCS puts the address of the record that is available for processing. For output files, IOCS puts the address of the area where the user can build a record. The same register may be used for different files.

This entry must be specified if blocked input or output records are processed in the I/O area or if two I/O areas are used and the records are processed in the I/O areas.

LABADDR=Name

Enter the symbolic name of the routine that enables the user to process his own labels. See the sections Writing and Checking User Standard DASD Labels for a discussion of what the LABADDR should do.

MODNAME=Name

This entry may be used to specify the name of the logic module that will be used to process the file. If the logic module is assembled with the program, the MODNAME in the DTF macro instruction must specify the same name as the SDMODxx macro instruction. If this entry is omitted, standard names will be generated for calling the logic module. If two DTF macro instructions call for different functions that can be handled by a single module, only one module is called.

NOTEPNT={POINTRW / YES}

The parameter POINTRW is specified if a NOTE, POINTR, or POINTW macro will be issued to the file. If the parameter YES is specified, NOTE, POINTR, POINTW, and POINTS macros may be issued to the file.

RECSIZE=n or (r)

For fixed-length blocked records, this entry is required. It specifies the number of characters, n, in each record.

For undefined records, this entry is required for output files and optional for input files. It specifies a general register (2-12) that contains the length of the record. On output, the user must load the length of each record into the register before he issues the PUT. If specified for input, IOCS will provide the length of the record transferred to main storage.

RECFORM=

This entry specifies the type of records (fixed or variable length, blocked or unblocked, or undefined) in the input or output file. One of the following may be entered immediately following the = sign.

FIXUNB for fixed-length unblocked records

FIXBLK for fixed-length block records

VARUNB for variable-length unblocked records

VARBLK for variable-length blocked records

UNDEF for undefined records.

If RECFORM is omitted, FIXUNB is assumed.

SEPASMB=YES

Include this parameter if the DTF is assembled separately. This causes a CATALR card with the filename to be punched ahead of the object deck and defines the filename as an ENTRY point in the assembly.

TRUNCS=YES

This entry is specified if FIXBLK DASD files contain short blocks embedded within an input file. This entry is also specified if the TRUNC macro will be issued for a FIXBLK output file.

TYPEFLE={INPUT / OUTPUT / WORK}

Use this entry to indicate whether the file is an input or output file. If WORK is specified, a work file is used. (See Work Files for DTFMT and DTFSD.) If INPUT is specified, the GET macro will be used. If OUTPUT is specified, PUT will be used. If WORK is specified, the READ/WRITE, NOTE/POINT, and CHECK macros will be used.

UPDATE=YES

This entry must be included if the DASD input file is to be updated.

VERIFY=YES

This entry is included if the user wants to check the parity of 2311 records after they are written. If this entry is omitted, any records written on 2311 are not verified. VERIFY is assumed when 2321 records are written.

VARBLD=(r)

Whenever variable-length blocked records are built directly in the output area (no work area specified), this entry must be included. It specifies the number (r) of a general-purpose register, which will always contain the length of the available space remaining in the output area. Any register 2-12 may be specified.

Only after the PUT instruction is issued for a variable-length record, IOCS calculates the space still available in the output area, and supplies it to the programmer in this VARBLD register. The programmer then compares the length of his next variable-length record with the available space to determine if the record will fit in the area. This check must be made before the record is built. If the record will not fit, the programmer issues a TRUNC instruction to transfer the completed block of records to the file. Then the present record is built at the beginning of the output area, as the first record in the next block.

WLRERR=Name

This entry applies to disk input files. It specifies the symbolic name of a user's routine to which IOCS will branch if a wrong-length record is read. In his routine the user can perform any operation he desires for wrong-length records. However, he must not issue any GET macro instructions for this file. Also, if he uses any other IOCS macros in his routine, he must save the contents of Register 14. The address of the wrong-length record is supplied by IOCS in Register 1. At the end of his routine, the user must return to IOCS by branching to the address in Register 14. When control is returned to the problem program, the first record of the next block is available for processing.

If this WLRERR entry is omitted from the set of DTFSD entries but a wrong-length record is detected by IOCS, one of the following will result:

- If the ERROPT entry is included for this file, the wrong-length record will be treated as an error block and handled according to the user's specifications for an error (IGNORE, SKIP, or Name of error routine).
- If the ERROPT entry is not included, the error will be ignored.

The WLRERR entry does not apply to undefined records. Undefined records are not checked for incorrect record length.

WORKA=YES

Input/output records can be processed, or built, in work areas instead of the input/output areas. If this is planned, this WORKA=YES entry must be included, and the programmer must set up the work area(s) in main storage. Then the symbolic name (or a general register containing the address), which is used in the DS instruction that reserves the work area, must be specified in each GET or PUT instruction. On a GET or PUT, IOCS moves the record to, or from, the specified work area.

Whenever this entry is included for a file, the DTF entry IOREG must be omitted.

PARAMETERS AND NAMES FOR SDMODxx SEQUENTIAL DASD MODULE

Sequential DASD module generation macros differ from other IOCS module generation macros. The file characteristics are separated into ten categories, and each category has a unique macro instruction associated with it.

Macro	Module Generated
SDMODFI	Sequential DASD Module, Fixed length records, Input file
SDMODFO	Sequential DASD Module, Fixed length records, Output file
SDMODFU	Sequential DASD Module, Fixed length records, Update file
SDMODVI	Sequential DASD Module, Variable length records, Input file
SDMODVO	Sequential DASD Module, Variable length records, Output file
SDMODVU	Sequential DASD Module, Variable length records, Update file
SDMODUI	Sequential DASD Module, Undefined records, Input file
SDMODUO	Sequential DASD Module, Undefined records, Output file
SDMODUU	Sequential DASD Module, Undefined records, Update file
SDMODW	Sequential DASD Module, Work file

As shown, the macro operation, as well as the keyword operands, define the characteristics of the module. Two advantages result from this way of generating modules for sequential DASD files:

1. Maintenance changes can be made to the module more easily.
2. A module to handle a specific file can be generated more quickly than if there were only one macro.

The operands for the ten macro instructions are shown in Figure 23 and explained in the following section.

SDMODxx Operands

A module name may be contained in the name field of the macro instruction. The macro operation is contained in the operation field (SDMODFI, for example). The operands are contained in the operand field.

CONTROL=YES This entry is specified if a CNTRL macro is issued to the file. This entry applies to all ten SDMODxx macro instructions. The module also processes any DTF in which the CONTROL parameter is not specified.

ERROPT=YES This entry applies to all ten SDMODxx macro instructions.

Name	Operation	Operand	Required	Comments
[Modname]	SDMODxx			
		CONTROL=YES	If the CNTRL macro is to be issued to the file	Applies to all SDMOD's
		ERROPT=YES	If the module will handle error options for an error block	Applies to all SDMOD's
		NOTEPNT={POINTRW / YES}	If NOTE, POINTR, POINTS, OR POINTW macros will be issued to the file	This parameter applies to SDMODW only. The operand POINTRW generates logic for NOTE, POINTR, and POINTW. The operand YES generates logic for all four macros.
		SEPASMB=YES	If the module is assembled separately from the DTF	Applies to all SDMOD's
		TRUNCS=YES	If the TRUNC macro will be issued to the file	Applies to all SDMOD's for fixed length records
		UPDATE=YES	If SDMODW will process the WRITE UPDATE macro instruction.	Applies to SDMODW only.

Figure 23. Parameters for SDMODxx

This entry is included if the module will handle any of the error options for an error block. Logic is generated to handle any of the three options (IGNORE, SKIP, or name) regardless of which option is specified. The module will process any DTF in which the ERROPT parameter is not specified.

If this entry is not included, the user's program will cancel when any uncorrectable error except wrong-length record error (which LIOCS will ignore) is encountered.

NOTEPNT={POINTRW / YES}

This entry applies to SDMODW (work files) only. This entry is included if any NOTE, POINTR, POINTS, or POINTW macro instructions are to be used with the module. If the operand specifies POINTRW, logic to handle only NOTE, POINTR, and POINTW is generated.

If the operand specifies YES, the routines to handle NOTE, POINTR, POINTS, and POINTW are generated and any files that specify NOTEPNT=POINTRW in the DTF are processed.

In either case, any files that do not specify the NOTEPNT parameter in the DTF are processed.

SEPASMB=YES Include this parameter if the logic module is assembled separately. This causes a CATALR card with the module name (standard or user) to be punched ahead of the object deck.

TRUNCS=YES This parameter applies to all SDMOD macro instructions for fixed length records. This entry causes generation of a logic module that allows use

of the TRUNC macro instruction. It must be specified if any FIXBLK DASD files to be processed by the module contain short blocks embedded within them. The module also processes any DTF, for fixed length records, in which the TRUNCS parameter is not specified.

UPDATE=YES This parameter is required for the SDMODW only. (It is assumed for SDMODFU, SDMODUU, and SDMODVU.) It causes generation of a logic module that will allow use of the WRITE UPDATE macro instruction with workfiles. (A DTFSD UPDATE entry is not required for workfiles, even though the SDMOD entry is required.)

Recommended Module Name List for SDMODxx

Each name will begin with a 3-character prefix (IJG) and consist of a 5-character field corresponding to the options permitted in the generation of the module.

In SDMOD there are two module classes: the module class for handling GET/PUT functions, and the module class for handling READ/WRITE, NOTE/POINT, and CHECK functions (work files).

Name List for GET/PUT Type Modules

SDMODxx name = IJGabcde

a = F if SDMODFx
  = V if SDMODVx
  = U if SDMODUx

b = U if SDMODxU
  = I if SDMODxI
  = O if SDMODxO

c = E if ERROPT=YES is specified
  = Z if ERROPT=YES is not specified

d = T if TRUNC=YES is specified
  = Z if TRUNC=YES is not specified

e = C if CONTROL=YES is specified
  = Z if CONTROL=YES is not specified

Name List for Workfile Type Modules (TYPEFLE=WORK)

SDMODxx name = IJGWabcd

a = E if ERROPT={IGNORE / SKIP / Name} is specified
  = Z if ERROPT is not specified

b = R if NOTEPNT=POINTRW is specified
  = N if NOTEPNT=YES is specified
  = Z if NOTEPNT is not specified

c = C if CONTROL=YES is specified
  = Z if CONTROL=YES is not specified

d = Z always

Subsetting and Supersetting of SDMOD Names

The following diagram illustrates the subsetting and supersetting allowed for SDMOD names. For the GET/PUT or workfile type modules, three parameters allow supersetting. For example, in the GET/PUT type module, the module IJGFUETC is a superset of a module with the name of IJGFUEZZ.

```
      **+++
  IJG FUETC
      VIZZZ
      UO
      *+++*
      WENCZ
       ZRZ
        Z

+Subsetting/supersetting permitted.
*No subsetting/supersetting permitted.
```

CONSOLE FILE (DTFCN)

DTFCN is used to define an input or output file that will be processed on an IBM 1052 Printer-Keyboard. DTFCN provides GET/PUT logic for the IBM 1052. The symbolic name of the file is entered in the name field and DTFCN is entered in the operation field. The detail entries, in any order, follow the DTFCN header entry with keyword entries in the operand field. Figure 24 contains the DTFCN entries.

When entering information on the printer-keyboard, if a mistake is made, press the CANCEL key. This causes a new read command to be issued, and the operator can retype the date starting from the beginning.

BLKSIZE=n

The length of the I/O area is stated in this entry. The length may be specified as a number n. If the record format is undefined, BLKSIZE must be as large as the largest record. The input/output records must not exceed 256 characters.

DEVADDR={SYSLOG / SYSnnn}

This entry specifies the symbolic unit that is associated with the logical file.

In a multiprogramming environment, DEVADDR=SYSLOG must be specified to obtain Background (BG), Foreground 1 (F1), or Foreground 2 (F2), prefixes for message identification.

IOAREA1=Name

This entry specifies the symbolic name of the I/O area to be used by the file.

RECFORM={FIXUNB / UNDEF}

This entry specified the record format of the file. FIXUNB is assumed.

RECSIZE=(r)

For undefined records this entry specifies the register (2-12) that will contain the length of the record. The user must load the length of each record into the register before issuing a PUT for the record on an output file.

TYPEFLE={INPUT / OUTPUT}

If INPUT is specified, coding will be generated for both input and output files. If OUTPUT is specified, coding for an output file only is provided.

WORKA=YES

This entry indicates that a work area will be used with the file. On a GET or PUT, IOCS moves the record to or from the work area.

OPERATION	OPERAND	MUST BE INCLUDED	REMARKS
DTFCN		For each file	Header entry . Specify 7-character or less symbolic filename in name field.
	BLKSIZE=n	For each file	Length of I/O area. n=actual length.
	DEVADDR= SYSLOG SYSnnn	For each file	Symbolic unit.
	IOAREA1= name	For each file	Specifies the symbolic name of the I/O area.
	RECFORM= FIXUNB UNDEF	For undefined files	Specifies the record format.
	RECSIZE=(r)	For each undefined file	Indicates register containing length of each record.
	TYPEFLE= INPUT OUTPUT	For each output file	Specifies the type of file.
	WORKA= YES		Specifies that a work area will be used.

Figure 24. DTFCN Entries

OPTICAL READER FILE (DTFOR)

DTFOR is used to define an input file to be processed on an IBM 1287 Optical Reader or on an IBM 1285 Optical Reader.

Enter the symbolic name of the file in the name field and DTFOR in the operation field. The entries for DTFOR are discussed in the following text and illustrated in Figure 25.

BLKSIZE=n

This entry indicates the size of the input area specified by IOAREA1. BLKSIZE specifies the maximum number (n) of characters that will be transferred to the area at any one time. When undefined records are read, the area must be large enough to accommodate the longest record to be read.

If two input areas are used for journal tape processing (IOAREA1 and IOAREA2), the size of only one area is specified in this entry.

IOCS uses this specification to construct the count field of the CCW.

CONTROL=YES

This entry must be included if a CNTRL macro instruction will be issued for this file. A control command issues orders to the Optical Reader to perform nondata operations such as line marking, stacker selecting, document incrementing, etc.

COREXIT=Name

COREXIT provides an exit to the user's error correction routine for the IBM 1285 or 1287 Optical Reader. Whenever the error correction routine is entered, an indication of the reason for the entry is provided in Filename+54. Filename+54 contains the following hexadecimal bits indicating the conditions that occurred during the last line or field read. More than one error condition may be present.

X'01' A reject character (@) was inserted in the line.

X'02' The operator corrected one or more characters from the keyboard.

X'04' A wrong-length record was retried unsuccessfully ten times. (Fixed-length unblocked records only.)

OPERATION	OPERAND	MUST BE INCLUDED	REMARKS
DTFOR		For each File	Header Card. Specify Symbolic File Name.
	BLKSIZE=n	For each File	Length of I/O Area.
	CONTROL=YES	If the CNTRL Macro is Used	
	COREXIT=name	For each File	Symbolic Name of Correction Routines.
	DEVADDR=SYSnnn	For each File	Symbolic Unit of I/O Device Used for File.
	DEVICE={1287T / 1287D / 1285}	Journal Tape Processing 1287 Document Processing 1287 1285 Processing	
	EOFADDR=name	For each File	Symbolic Name of End-of-File Routine.
	HEADER=YES	If a Header Record is to be Read from Keyboard by OPEN	If Omitted, OPEN Assumes no Header Record.
	IOAREA1=name	For each File	Symbolic Name of Input Area.
	IOAREA2=name	If two I/O Areas are Used	Omit for 1287 Document Processing.
	IOREG=r	If two I/O Areas are Used	r = Number of a General Purpose Register (2-12). Omit WORKA=YES
	RECFORM={FIXUNB / UNDEF}	For each File	Specify Record Format.
	RECSIZE={n / r}	Fixed Length Unblocked Records Undefined Records	n = Number of Characters in the Record r = Register 2-12
	TYPEFILE=INPUT	For each File	
	WORKA=YES	If GET Specifies a Work Area	Omit if IOREG=r.

Figure 25. DTFOR Entries

X'08' An equipment check resulted in a lost line.

X'10' A non-recovery error occurred.

X'20' A stacker-select command was given after the allotted time had elapsed and the document had been put in the reject pocket.

Filename+54 can be interrogated by the user to determine the reason for entry to the error correction routine. Choice of action in the user's error correction routine is determined by the particular application. Only the entries pertaining to wrong-length record, late stacker selection, equipment check and non-recovery are applicable to document processing on the 1287.

If the user issues the CNTRL, DSPLY, RDLNE, or RESCN macros in his error routine within COREXIT he must first save, and later restore registers 14 and 15. All exits, except as noted below from COREXIT, must be to the address in register 14, which returns to the point from which the branch to COREXIT occurred.

Seven binary counters are used to accumulate totals of certain 1285 and 1287 tape mode error conditions. These counters each occupy four bytes, starting at Filename+22.

Filename is the name specified in the DTFOR header entry. The seven counters are:

Counter	Address	Contents
1	Filename+22	Lost lines
2	Filename+26	Lost line uncorrectable after ten retries.
3	Filename+30	Wrong length records
4	Filename+34	Wrong length records uncorrectable after ten retries.
5	Filename+38	Keyboard corrections
6	Filename+42	Reject characters
7	Filename+46	Lines marked

An eighth counter, located at Filename+50, contains a count of total lines read. It is cleared at the start of each job run.

The user may list the contents of these counters for analysis at end of file, or at end of job, or he may ignore the counters. (Binary contents of the counters should be converted to a printable format.)

A non-recoverable error when processing journal tapes (torn tape, tape jam, etc) may require that the tape be completely reprocessed unless the user can provide suitable means of correction. A non-recovery, when processing documents on the IBM 1287 Optical Reader, indicates that a jam occurred during a document incrementation operation. It may or may not indicate loss of the document, depending on the extent of document mutilation. It is recommended in this case that the problem program not exit to the address in register 14 from COREXIT. The exit should be made to the point in the problem program at which a new document is read. The user must ignore any output resulting from the document which caused the jam.

Notes: The user cannot issue a GET or a READ in his error correction routine. The user must not process records in his error correction routine. The record that caused the exit to the error routine will be available for processing upon return to the user's mainline program. Any processing included in the error routine would be duplicated after return to the mainline program.

DEVADDR=SYSnnn

This entry specifies the symbolic unit (SYSnnn) to be associated with this logical file. The symbolic unit represents an actual I/O device address. The symbolic unit is SYS000-SYS244 for an optical reader. It is used in the Job Control ASSGN statement to assign the actual I/O device address to this file.

DEVICE=

This entry must be included to state the I/O device associated with this logical file. One of the following specifications must be entered immediately after the = sign:

1287T	for a journal tape input file on a 1287
1287D	for a document input file on a 1287
1285	for a journal tape input file on a 1285

From this specification, IOCS sets up the proper channel command words (CCW) and device-dependent routines for this file. For document processing on the 1287 Optical Reader, the coding of CCW's is accomplished by the user allowing command chaining, comparing, and branching without experiencing an interrupt for each function.

EOFADDR=Name

This entry specifies the symbolic name of the user's end-of-file routine. IOCS automatically branches to this routine on an end-of-file condition. In his routine, the programmer can perform any operations required for the end of the file, and he generally issues the CLOSE instruction for the file.

When processing documents, end-of-file condition is recognized by pressing the End-of-File key on the console when the hopper is empty. When processing journal tapes on a 1285 or 1287, end-of-file is detected by pressing the End-of-File key after the end of the tape has been sensed.

When IOCS detects the end-of-file, it branches to the user's routine specified by EOFADDR. It is the user's responsibility to determine if the current roll is the last roll to be processed. The tape file on the 1285 or 1287 must be closed with the CLOSE macro in the end-of-file routine. If the current roll is not the last, OPEN must be issued. The OPEN macro instruction allows header (identifying) information to be entered at the reader keyboard and read by the processor when using logical IOCS.

HEADER=YES

This entry is required if the operator is to key in header (identifying) information from the 1285 or 1287 keyboard. The OPEN routine reads the header information only when this entry is present. If the entry is not included, OPEN assumes no header information is to be read.

IOAREA1=Name

This entry is included to specify the symbolic name of the input area used by this file. The input/output routines will transfer records to this area.

IOAREA2=Name

Two input areas can be allotted for a file to permit an overlap of data transfer and processing operations. When this is done, the IOAREA2 entry must be included. It specifies the symbolic name of the second I/O area.

This entry must not be included if DEVICE=1287D.

IOREG=(r)

This entry specifies the general-purpose register (r) that the input/output routines use to indicate the beginning of undefined records for an optical reader journal tape file using two input areas. Any register number 2-12 may be specified.

The same register may be specified in the IOREG entry for two or more files in the same program, if desired. In this case, the problem program may need to store the address supplied by IOCS for each record.

Whenever this entry is included for a file, the DTFOR entry WORKA must be omitted, and the GET instruction must not specify a work area.

Undefined records are processed by the 1285 and 1287. The "read" by an optical reader is accomplished by a backward scan, which places the rightmost character in the record in the rightmost position in the I/O area and subsequent characters in sequence from right to left. The register defined by IOREG is used to indicate, to the user, the beginning of the record.

RECFORM=

This entry specifies the type of records (fixed unblocked or undefined) in an optical reader file. One of the following specifications may be entered immediately after the = sign:

FIXUNB	for fixed-length unblocked records
UNDEF	for undefined records

RECSIZE=n or (r)

For fixed-length unblocked records, this entry specifies the number n of characters in a record. The input/output routines use this factor to check record length of input records.

For undefined records, this entry specifies the number (r) of the general-purpose register that will contain the length of each individual input record. This may be any register 2-12. When undefined optical reader records are read, IOCS supplies the physical record size in the register. If IOAREA2 is specified, a second register for this purpose is specified by an IOREG entry.

TYPEFLE=INPUT

This entry must be included to specify that the optical reader file is an input file.

WORKA=YES

Journal tape input records can be processed in a work area instead of the input area. If this is planned, the entry WORKA=YES must be included, and the programmer must set up the work area in main storage. The symbolic name used in the DS instruction that reserves the work area (or a general register containing the address of the work area) must be specified in each GET. When GET is issued, IOCS moves the record to the specified work area.

Whenever this entry is included for a file, the DTFOR entry IOREG must be omitted.

SPECIAL CONSIDERATIONS FOR OPTICAL READER FILES

Record Formats

Two record formats, fixed unblocked and undefined, are supported for both modes of 1287 processing and for 1285 processing.

FIXED UNBLOCKED RECORDS: This format is used when processing journal tapes which have an equal number of characters in each

line. However, each field to be read from a document may be treated as fixed or variable in length by setting the suppress length indicator bit in the CCW used to read it. Therefore, the description of a document as fixed-unblocked does not have its usual meaning. It is provided for the user who does not wish to use the facilities of RECSIZE and IOREG with undefined records. This eliminates the requirement for these two registers. Except for this feature, record processing is unchanged from processing of records in the undefined format. RECSIZE should contain the length of the input areas.

UNDEFINED RECORDS: These records are to be used when processing journal tapes with lines of variable character length and documents as described under Fixed Unblocked Records. The information supplied by the DTFOR entries, RECSIZE and IOREG, pertains only to the last field of a document read by the user-supplied channel command word chain when used with documents specified as undefined.

DATA ACQUISITION AND MANIPULATION: Data is acquired from journal tapes with the GET macro instruction. Processing overlap is obtained by using multiple input areas. Data is obtained from documents with the READ macro instruction.

READ generates an EXCP and a CCB macro which cause a branch to the user-provided channel command chain. When processing documents, only one input area can be used. The contents of the input area may then be moved to a user-defined work area. Processing overlap may be obtained by processing the contents of the work area while the next document is being read into the input area. Another method of obtaining processing overlap is to divide the fields on a document into blocks and process one block while reading another.

WRONG LENGTH RECORD CONDITIONS: Only wrong length record conditions which occur when processing fixed unblocked fields on a document or when processing fixed unblocked records from a journal tape can be detected. When processing undefined records on fields, the SLI flag must be set on.

REFERENCE MARKS: At least one reference mark is required for all documents. In addition, new co-ordinates for the reference mark for a document must be specified whenever a rotation in printing occurs and whenever a document is incremented. For this reason, a load format CCW which specifies the co-ordinates of the reference mark associated with a particular group of fields must be the first CCW in all user channel command word chains.

SERIAL DEVICE FILE (DTFSR)

The DTFSR macro instruction is provided as a compatibility aid to users of the Basic Operating System/360. For Disk Operating System/360 users, the DTFCD, DTFPT, DTFMT, DTFOR, DTFPR, DTFCN, and DTFSD macros provide more advantages than DTFSR. Enter the symbolic name of the file in the name field and DTFSR in the operation field. The entries for DTFSR are discussed in the following text and illustrated in Figure 25.

The begin-definition card must be punched with DTFBG in the operation field and DISK in the operand field. The name field is blank. It is included in DOS to provide compatability with the BOS DTFSR macro instruction.

ALTTAPE=SYSnnn

This parameter is provided for BPS and BOS compatibility.

BLKSIZE=n

This entry indicates the size of the input, or output, area specified by IOAREA1. BLKSIZE specifies the maximum number (n) of characters that will be transferred to, or from, the area at any one time. When variable-length records are read, or written, the area must be large enough to accommodate the largest block of records, or the longest single record if the records are unblocked.

If card-punch or printer output records include control characters (DTFSR CTLCHR specified) and/or record-length fields for variable-length records (RECFORM=VARUNB), the BLKSIZE specification must include the extra bytes allotted in the main-storage output area.

If two input, or output, areas are used for a file (IOAREA1 and IOAREA2), the size of only one area is specified in this entry.

OPERATION	OPERAND #	APPLIES TO									MUST BE INCLUDED	REMARKS *
		2311 DISK DRIVE	2400 MAGNETIC TAPE UNIT	1442/2501/2520/ 2540 READER	1442/2520/2540 PUNCH	1403/1404/1443/ 1445 PRINTER	1052 PRINTER-KEYBOARD	2671 PAPER TAPE READER	1285 OPTICAL READER	1287 OPTICAL READER		
DTFSR†		X	X	X	X	X	X	X	X	X	For each file.	Header entry. Specify 7-character or less symbolic filename in the name field.
	ALTTAPE=SYSnnn		X								Multivolume file using two tape drives	Symbolic unit for alternate tape drive. (For compatibility purposes)
	BLKSIZE=n	X	X	X	X	X	X	X	X	X	Each file except combined file with separate I/O areas	Length of I/O area. n=maximum number of characters.
	CHECKPT=n		X								CHKPT macro used	(For compatibility purposes)
	CKPTREC=YES		X								Bypass checkpoint records on input	Applies to input file only.
	CONTROL=YES	X	X	X	X	X			X	X	CNTRL macro used	CTLCHR must be omitted. Does not apply to 2501.
	COREXIT								X	X	For correction routines	Symbolic name of correction routine
	CRDERR=RETRY				X						Punch again on error condition	Applies only to 2540 and 2520
	CTLCHR=YES				X	X					Logical records have control character in first position	Each record must contain a control character. CONTROL must be omitted and the CNTRL macro must not be used for this file.
	DEVADDR=SYSnnn		X	X	X	X	X	X	X	X	For each file except disk because an XTENT card supplies this data	Symbolic unit for the I/O device used for the file.
	†DEVICE=DISK11	X									Disk input/output file	Include for each file, and specify proper name after DEVICE=.
	TAPE		X								Tape input/output file	
	READ01			X							2501 input file	
	READ20			X	X						2520 input/output file	
	READ40			X	X						2540 input/output file	
	READ42			X	X						1442 input/output file	
	PRINTER					X					1403,1404,1443,1445 output	
	CONSOLE						X				1052 input/output	
	PTAPERD							X			2671 input file	
	READ87T									X	1287 tape input file	
	READ87D									X	1287 document input file	
	READ85								X		1285 input file	
	EOFADDR=Name	X	X	X				X	X	X	For an input file	Symbolic name of end-of-file routine.
	ERROPT=IGNORE	X	X					X			To process error records	Applies only to disk or tape input. Prevents job termination on error condition. Enter desired specification after ERROPT=.
	SKIP										To skip over error records	
	Name										For user routine for error records	
	FILABL=STD		X								To check or write standard labels	Include for tape input/output and specify desired operation after FILABL=.
	NSTD										For a file containing nonstandard labels	
	NO										For an unlabelled file	
	HEADER=YES								X	X	For a header record to be read by OPEN	If this entry is omitted, OPEN assumes no header record.
	INAREA=Name			X							To separate areas for input and output for a combined file	Applies only to 1442.
	INBLKSZ=n			X							To separate areas for input and output for a combined file	Applies only to 1442. Length of INAREA.

Figure 26. DTFSR Entries (Part 1 of 3)

OPERATION	OPERAND *	APPLIES TO									MUST BE INCLUDED	REMARKS *
		2311 DISK DRIVE	2400 MAGNETIC TAPE UNIT	1442/2501/2520/ 2540 READER	1442/2520/2540 PUNCH	1403/1404/1443/ 1445 PRINTER	1052 PRINTER-KEYBOARD	2671 PAPER TAPE READER	1285 OPTICAL READER	1287 OPTICAL READER		
DTFSR†	IOAREA1 = Name	X	X	X	X	X	X	X	X	X	For each file except 1442 combined file with separate I/O areas	Symbolic name of input/output area. I/O Area for 2520 or 2540 Combined Files
	IOAREA2 = Name	X	X	X	X	X	X	X	X	X	For two I/O areas used	Symbolic name of second input/ output area.
	IOREG = (r)	X	X	X	X	X	X	X	X	X	To process blocked records in I/O area, or process in two I/O areas	r = number of general purpose register 2-12. Omit WORKA = YES.
	LABADDR = Name	X	X								To check/build additional user-standard labels, or process nonstandard labels	Symbolic name of user's label routine.
	OUAREA = Name			X							To separate areas for input and output for a combined file	Applies only to 1442.
	OUBLKSZ = n			X							To separate areas for input and output for a combined file	Applies only to 1442. Length of OUAREA.
	PRINTOV = YES					X					For PRTOV macro used	
	READ = FORWARD		X									If this entry omitted, IOCS assumes FORWARD.
	BACK										To read tape backwards	
	RECFORM = FIXUNB	X	X	X	X	X	X	X	X	X		Specify as needed for fixed-length unblocked records. If this entry omitted, IOCS assumes FIXUNB.
	FIXBLK	X	X								For fixed-length blocked records	
	VARUNB	X	X		X	X					For variable-length unblocked records	Disk or tape records require record-length field.
	VARBLK	X	X								For variable-length blocked records	Blocks require block-length field. Records require record-length field.
	UNDEF	X	X		X	X	X	X	X	X	For undefined records	
	RECSIZE = n or (r)	X	X		X	X	X	X	X X	X X	For fixed-length blocked records For undefined records	n = number of characters in record. r = number of register 2-12.
	REWIND = UNLOAD		X								To unload on CLOSE or end-of-volume	Omit to rewind only at OPEN, CLOSE, or end-of-volume.
	NORWD										To prevent rewinding	
	TPMARK = NO		X								If FILABL = NO or NSTD and no tape mark is desired	If omitted, YES is assumed.
	TRANS = Name							X			For tape punched with code other than EBCDIC and IOCS is to perform translation	Symbolic name of code translation table.
	TRUNCS = YES	X									For fixed-length blocked records with short blocks	Include for output if TRUNC macro used. Include for input if TRUNC macro was used when file was created.
	TYPEFLE = INPUT	X	X	X			X	X	X	X	For each file INPUT is assumed	Specify proper type after TYPEFLE = CMBND applies to 1442, 2520, or to 2540 if punch-feed-read special feature is installed.
	OUTPUT											
	CMBND											
	UPDATE = YES	X									For PUT to be used for a disk input file	
	VARBLD = (r)	X	X								For variable-length blocked records built in output area	r = number of a register 2-12.
	VERIFY = YES	X									To check record written on disk	

Figure 26. DTFSR Entries (Part 2 of 3)

OPERATION	OPERAND *	APPLIES TO									MUST BE INCLUDED	REMARKS *
		2311 DISK DRIVE	2400 MAGNETIC TAPE UNIT	1442/2501/2520/2540 READER	1442/2520/2540 PUNCH	1403/1404/1443/1445 PRINTER	1052 PRINTER-KEYBOARD	2671 PAPER TAPE READER	1285 OPTICAL READER	1287 OPTICAL READER		
DTFSR†	WLRERR=Name	X	X					X			For user routines for wrong-length records	Symbolic name of user's routine. If omitted, error handled as in ERROPT; or if both entries omitted; WLR indication will be ignored.
	**WORKA=YES	X	X	X	X	X	X		X	X‡	For GET or PUT to specify a work area	Omit IOREG=(r)

† Must be included. Other entries are included when applicable.

When two or more choices are shown, select only the appropriate one and enter it after the = sign.

* The header and each detail card except the last one used in a file set must contain a continuation punch in column 72. Each detail card except the last one used must also contain a comma after the last operand.

** The 16K DTFSR does not support a work area when the 2671 Paper Tape Reader is specified as the device.

In all entries: Solid caps must be entered as shown (For example, CONTROL=YES).

Lowercase letters are to be replaced by programmer's symbolic name or a number (For example, Filename in header card, or BLKSIZE=n where n is replaced).

n is a decimal self-defining value.

‡ This entry should not be used if DEVICE=1287D.

Figure 26. DTFSR Entries (Part 3 of 3)

IOCS uses this specification to:

- Construct the count field of the CCW for an input file.
- Construct the count field of the CCW for an output file of fixed-length records.
- Check physical record length for a file of fixed-length blocked input records.
- Determine if the space remaining in the output area is large enough to accommodate the next variable-length output record.

CHECKPT=n

This entry is provided for compatibility with BPS and BOS and is ignored by DOS.

CKPTREC=YES

This entry is required if a tape input file will contain checkpoint records interspersed among the data records. With this entry, IOCS recognizes the checkpoint records and bypasses them.

CONTROL=YES

This entry must be included if a CNTRL macro instruction will be issued for this file. A control command issues orders to the I/O device to perform non-data operations such as card-stacker selection, carriage skipping, marking of doubtful 1287 or 1285 tape records, tape rewinding, etc.

When CONTROL is included, the DTFSR entry CTLCHR must not be included.

COREXIT=NAME

COREXIT provides an exit to the user's error correction routine for the IBM 1285 or 1287 Optical Reader. Whenever the error correction routine is entered, an indication of the reason for the entry is provided in Filename+54. Filename+54 contains the following hexadecimal bits, indicating the conditions that occurred during the last line or field read. More than one error condition may be present:

X'01' A reject character (@) was inserted in the line.

X'02' The operator corrected one or more characters from the keyboard.

X'04' A wrong-length record was retried unsuccessfully ten times. (Fixed-length unblocked records only.)

X'08' An equipment check resulted in a lost line.

X'10' A non-recovery error occurred.

X'20' A stacker-select command was given after the allotted time had elapsed and the document had been put into the reject pocket.

Filename+54 can be interrogated by the user to determine the reason for entry to the error correction routine. Choice of action in the user's error correction routine is determined by the particular application. Only the entries pertaining to wrong-length record, late stacker selection, equipment check and non-recovery are applicable to document processing on the 1287.

If the user issues the CNTRL, DSPLY, RDLNE, or RESCN macros in his error routine within COREXIT, he must first save, and later restore registers 14 and 15. All exits except as noted below from COREXIT must be to the address in register 14, which returns to the point from which the branch to COREXIT occurred.

Seven binary counters are used to accumulate totals of certain 1285 and 1287 tape mode error conditions. These counters each occupy four bytes, starting a Filename+22. Filename is the name specified in the DTFOR header entry. The seven counters are:

Counter	Address	Contents
1	Filename+22	Lost lines.
2	Filename+26	Lost lines uncorrectable after ten retries.
3	Filename+30	Wrong-length records.
4	Filename+34	Wrong-length records uncorrectable after ten retries.
5	Filename+38	Keyboard corrections.
6	Filename+42	Reject characters.
7	Filename+46	Lines marked.

An eighth counter, located at Filename+50, contains a count of total lines read. It is cleared at the start of each job run.

The user may list the contents of these counters for analysis at end of file, or at end of job, or he may ignore the counters. (Binary contents of the counters should be converted to a printable format.)

A non-recovery error when processing journal tapes (torn tape, tape jam, etc) may require that the tape be completely reprocessed unless the user can provide a suitable means of correction. A non-recovery, when processing documents on the IBM 1287 Optical Reader, indicates that a jam occurred during a document incrementation operation. It may or may not indicate loss of the document, depending on the extent of document mutilation. It is recommended in this case that the problem program not exit to the address in register 14 from COREXIT. The Exit should be made to the point in the problem program at which a new document is read. The user must ignore any output resulting from the document which caused the jam.

Notes: The user cannot issue a GET or a READ macro in his error correction routine.

The user must not process records in his error correction routine. The record that caused the exit to the error routine will be available for processing upon return to the user's mainline program. Any processing included in the error routine would be duplicated after return to the mainline program.

See Special Considerations for Optical Reader Files in the Optical Reader File (DTFOR) section of this manual.

CRDERR=RETRY

This entry applies only to a card output file in the IBM 2540 or 2520. It specifies the operation to be performed if an error is detected.

Normally if a punching error occurs, it is ignored and operation continues. The error card is stacked in pocket P1 (punch). Correct cards are stacked in the pocket selected by the user. If this CRDERR entry is included to specify retrying, however, IOCS also notifies the operator and then enters the wait state when an error condition occurs. The operator can either terminate the job or instruct IOCS to repunch the card.

From this specification, IOCS generates a retry routine and a save area for the card punch record.

CTLCHR=YES

The CTLCHR (control character) entry applies only to printer and punch output files. It is included if each logical record to be written or punched contains a control character (carriage control or stacker selection) in the record itself, in the main-storage output area. For fixed-length or undefined records, the control character must be the first character. For variable-length records, it is the first character after the record-length field. The control character codes are the same as the modifier bytes used for a punch or print command.

With this entry, the IOCS routines cause the control-character-specified printer or card punch order to be issued to the I/O device. Printing or punching begins with the second character in the record.

When the CTLCHR entry is not included, any control functions desired must be performed by the CNTRL macro.

DEVADDR=SYSnnn

This entry specifies the symbolic unit (SYSnnn) to be associated with this logical file. The symbolic unit represents an actual I/O device address. The symbolic unit for the batch-or background job may be:

SYSRDR for main system control card reader

SYSIPT for main system input device

SYSPCH or SYSOPT for main system punch device

SYSLST for main system printer

SYSLOG for control card logging device

SYS000-SYS244 for other units in the system. This is generally a unique number for each logical file (except files on disk).

For the foreground job, the symbolic units available are: SYSLOG, SYS000-SYSnnn (no more than 245, total, symbolic units with numeric names can exist in a system).

The symbolic unit (SYSnnn) is used in the Job Control ASSGN statement to assign the actual I/O device address to this file. A reel of tape may be mounted on any tape drive that is available at the time the job is ready to be run, by merely assigning that drive to the symbolic unit.

Whenever two devices are used for one logical file, such as an alternate tape drive (specified in the ASSGN cards), this DEVADDR entry specifies the symbolic unit for the first device.

The symbolic unit is specified for all units except the 2311 disk drive. For files on this unit, DEVADDR may be omitted. If DEVADDR is omitted, the symbolic unit for a disk drive is supplied by a Job Control XTENT card.

DEVICE=

This entry must be included to state the I/O device associated with this logical file. One of the following specifications must be entered immediately after the = sign.

DISK11	for an input or output file on disk (2311).
TAPE	for an input or output file recorded on magnetic tape (2401, 2402, 2403, 2404, 2415).
PRINTER	for output printed on a 1403, 1404, 1443, or 1445.
READ01	for an input card file in a 2501.
READ20	for an input or output card file in a 2520.
READ40	for an input or output card file in a 2540.
READ42	for an input or output card file in a 1442.
CONSOLE	for input from and output to the printer-keyboard (1052).
PTAPERD	for input from a 2671.
READ87T	for a journal tape input file on a 1287
READ87D	for a document input file on a 1287
READ85	for a journal tape input file on a 1285

From this specification, IOCS sets up the proper channel command words (CCW's) and device dependent routines for this file. For document processing on the 1287 Optical Reader, the coding of CCW's is accomplished by the user allowing command chaining, comparing, and branching without experiencing an interrupt for each function.

EOFADDR=Name

This entry must be included for:

- Card reader files
- Magnetic tape input files
- Paper tape input files
- Sequential disk input files.
- Optical Reader files.

It specifies the symbolic name of the user's end-of-file routine. IOCS will automatically branch to this routine on an end-of-file condition. In his routine, the programmer can perform any operations required for the end of the file, and he generally issues the CLOSE macro instruction for the file.

IOCS detects end-of-file conditions as follows:

- Card reader--by recognizing /* punched in card columns 1 and 2 if the reader is assigned to a programmer unit, or by recognizing either /* or /& if the unit is assigned to SYSRDR or SYSIPT. If cards are allowed to run out without a /* or a /& trailer card, an error condition is signaled to the operator (intervention required).
- Magnetic tape input--by reading a tapemark and EOF in the trailer label when standard labels are specified, or by reading /* if the unit is assigned to SYSRDR or SYSIPT. If standard labels are not specified, IOCS assumes an end-of-file condition when the tapemark is read. The user must determine, in his routine, that this actually is the end of the file.
- Paper tape reader--by recognizing the end of tape when the end-of-file switch is set ON.
- Sequential disk input--by reading an end-of-file record or reaching the end of the last extent supplied by the user.
- 1287 Optical Reader input--when processing documents, end-of-file condition is recognized by depression of the End-of-File key on the console when the input hopper is empty. When processing journal tapes on a 1287 or 1285, end-of-file is detected by depression of the End-of-File key after the end of the tape has been sensed.

When IOCS detects the end of file, it branches to the user's routine specified by EOFADDR. It is the user's responsibility to determine if the current roll

is the last roll to be processed. The tape file on the 1287 or 1285 must be closed with the CLOSE macro in the end-of-file routine. If the current roll is not the last, OPEN must be issued. The OPEN macro instruction allows header (identifying) information to be entered at the reader keyboard and read by the processor when using logical IOCS.

ERROPT=

This entry applies to disk or magnetic tape input files, and it specifies functions to be performed for an error block.

If a parity error is detected when a block of sequential disk records is read, the disk block is reread 10 times before it is considered an error block. If a parity error is detected when a block of tape records is read, the tape is backspaced and reread 100 times before the tap block is considered an error block. After this the job is automatically terminated, unless this ERROPT entry is included to specify other procedures to be followed on an error condition. Either IGNORE, SKIP, or the symbolic name of an error routine can be specified in this card. One of these specifications is entered immediatel after the = sign in this keyword operand.

The functions of these three specifications are:

IGNORE — The error condition is completely ignored, and the records are made available to the user for processing.

SKIP — No records in the error block are made available for processing. The next block is read from disk or tape, and processing continues with the first record of that block. The error block is included in the block count, however.

Name — IOCS branches to the user's routine, where he may perform whatever functions he desires to process or make note of the error condition. Register 1 contains the address of the block in error, and register 14 contains the return address.

In his routine, the programmer should address the error block, or records in the error block, by referring to the address supplied in register 1. The contents of the IOREG register or the work area (if either is specified) may vary and, therefore, should not be used for error blocks. Also, the programmer must not issue any GET instructions for records in the error block. If he uses any other IOCS macros in his routine, he must save the contents of register 14. At the end of his routine, the programmer must return to IOCS by branching to the address in register 14. When control is returned to the problem program, the first record of the next block is available for processing in the main program.

This ERROPT entry does not apply to disk or tape output files. The job is automatically terminated if a parity error still exists after IOCS attempts 10 times to write a disk output block, or 15 times to write a tape output block. The tape procedure includes 15 forward erases.

This entry applies to wrong-length records if the DTFSR entry WLRERR is not included. If both ERROPT and WLRERR are omitted, IOCS ignores any wrong-length records that occur.

FILABL={NO / STD / NSTD}

This entry may be included for a tape input or output file. One of the following specifications is entered immediately after the = sign:

STD — for a tape input file if standard labels are to be checked by IOCS, or for a tape output file if standard labels are to be written by IOCS.

NSTD — for a tape input or output file that has nonstandard labels. These labels may be processed by the user (see Opening Tape Files: Nonstandard Labels). NSTD is specified for standard input labels if they are not to be checked by IOCS.

NO — for a tape file that does not contain labels. The entry FILABL=NO may be omitted, if desired, and IOCS will assume that there are no labels.

HEADER=YES

This entry is required if the operator is to key in header (identifying) information from the 1285 or 1287 keyboard. The OPEN routine reads the header information only when this entry is present. If the entry is omitted, OPEN assumes no header information is to be read.

INAREA=Name

This entry applies only to a card file in an IBM 1442 that is to be updated (TYPEFLE=CMBND) and for which separate input and output areas are required. INAREA specifies the symbolic name of the input area to which the card record is to be transferred. OUAREA is used in conjunction with INAREA, and both IOAREA1 and IOAREA2 must be omitted.

When the same I/O area is to be used for both input and output in a combined file for a 2520 or 2540, INAREA and OUAREA are omitted, and IOAREA1 specifies the name of the I/O area to be used for both input and output.

INBLKSZ=n

This entry is used in conjunction with INAREA for a combined file in the 1442 when separate input and output areas are required. It specifies the maximum number, n, of characters that will be transferred to the input area (INAREA) at any one time. Whenever this entry is included, the corresponding entry OUBLKSZ must also be included, and BLKSIZE must be omitted.

IOAREA1=Name

This entry is included to specify the symbolic name of the input, or output, area used by this file. The input/output routines will transfer records to or from this area.

For a disk output file, the user must reserve eight bytes at the beginning of his I/O area, ahead of the positions allotted for data records. These eight bytes are necessary to allow IOCS to construct the count area for the disk record.

This entry must not be included for a 1442 combined file if INAREA and OUAREA are specified for the file.

For a 2520 or 2540 combined file, IOAREA1 must be used for both the input and output area.

IOAREA2=Name

Two input, or output, areas can be allotted for a file, to permit an overlap of data transfer and processing operations. When this is done, this IOAREA2 entry must be included. It specifies the symbolic name of the second I/O area.

For a disk output file, the user must reserve eight bytes at the beginning of his I/O area, ahead of the positions allotted for data records. These eight bytes are necessary to allow IOCS to construct the count area for the disk record.

This entry must not be included for a 1442 combined file if INAREA and OUAREA are specified for the file, if TYPEFILE=CMBND, or if DEVICE=READ870.

For a 2520 or 2540 combined file, IOAREA2 cannot be specified. IOAREA1 in this case must be used for both the input and output areas.

IOREG=(r)

This entry specifies the general-purpose register (r) that the input/output routines can use to indicate which individual record is available for processing. IOCS puts the address of the current record in this register each time a GET or PUT is issued. Any register number 2-12 may be specified.

The same register may be specified in the IOREG entry for two or more files in the same program, if desired. In this case the problem program may need to store the address supplied by IOCS for each record.

This entry must be included whenever:

- Blocked input or output records (from disk or tape) are processed directly in the I/O area.
- Variable-length unblocked or undefined tape records are read backwards and processed directly in the input area.
- Two input, or output, areas are used and the records (either blocked or unblocked) are processed in the I/O areas.

Whenever this entry is included for a file, the DTFSR entry WORKA must be omitted and the GET, or PUT, instructions must not specify work areas.

Undefined records are processed by the IBM 1287 Optical Reader and the IBM 1285 Optical Reader. The "read" by the optical reader is accomplished by a backward scan, which places the rightmost character in the record in the rightmost position in the I/O area and subsequent characters in sequence from right to left. The register defined by IOREG is used to indicate, to the user, the beginning of the record.

LABADDR=Name

The user may require one or more of his own disk or tape labels in addition to the standard file header label or trailer label (on tape). If so, he must include his own routine to check, or build, the label(s). The symbolic name of his routine is specified in this entry. IOCS branches to this routine after it has processed the standard label. This entry is also required whenever nonstandard labels are to be checked or written by the user (DTFSR FILABL specifies NSTD).

LABADDR allows one user's label routine to be specified for all types of labels for the file: header labels, end-of-file labels, and end-of-volume labels. On an input file, the user can determine the type of label that has been read by the identification in the label itself. For an output tape file, however, IOCS indicates to the user the type of label that is to be

written. For this, IOCS supplies a code in the low-order byte of Register 0, as follows:

O--Header label
F--End-of-file label
V--End-of-volume label

In his routine the user can test this byte and then build the appropriate type of label.

At the end of his routine, the programmer must return to IOCS by use of the LBRET macro. The user may not issue a macro instruction that will call in a transient routine. For example, the OPEN, CLOSE, DUMP, PDUMP, CANCEL, or CHKPT macro cannot be issued in the LABADDR routine.

OUAREA=Name

This entry is used in conjunction with INAREA for a combined file in an IBM 1442 that requires separate input and output areas. It specifies the symbolic name of the output area from which the updated card record is punched. If only one area is to be used for input and output, then IOAREA1 should be used.

OUBLKSZ=n

This entry is used in conjunction with OUAREA for a combined file. Similar to INBLKSZ, it specifies the maximum number, n, of characters that will be transferred from the output area (OUAREA) at any one time. If combined files use IOAREA1, then BLKSIZE must be used.

PRINTOV=YES

This entry must be included whenever the PRTOV macro instruction is included in the problem program.

READ=

This entry may be included for a tape input file to specify the direction in which the tape is to be read. One specification or the other is entered immediately after the = sign:

FORWARD for a tape read in the normal forward direction.

BACK for a tape read backwards.

If this entry is omitted, IOCS assumes forward reading.

RECFORM=

This entry specifies the type of records (fixed or variable length, blocked or unblocked, or undefined) in the input or output file. One of the following specifications may be entered immediately after the = sign:

FIXUNB for fixed-length unblocked records.

FIXBLK for fixed-length blocked records. This applies only to disk and magnetic tape input or output.

VARUNB for variable-length unblocked records. This applies only to disk input or output (2311), magnetic tape input or output (2400), card punch output (1442, 2520, or 2540), and printer output (1403, 1404, 1443, or 1445).

VARBLK for variable-length blocked records. This applies only to disk and magnetic tape input or output.

UNDEF for undefined records. This applies to any file except card input (1442, 2501, 2520, or 2540).

Thus the records in a file can be specified as follows:

Disk and magnetic tape input or output:
FIXUNB, FIXBLK, VARUNB, VARBLK, or UNDEF

Card input: FIXUNB

Card output: FIXUNB, VARUNB, or UNDEF

Printer output: FIXUNB, VARUNB, or UNDEF

Printer-keyboard input or output: FIXUNB or UNDEF

Paper tape input: FIXUNB or UNDEF

RECSIZE=n or (r)

This entry must be included for disk or magnetic tape records that are fixed-length blocked (RECFORM=FIXBLK) or undefined (RECFORM=UNDEF), in an input or output file. For paper tape records, this entry may be included for fixed-length unblocked or for undefined records (RECFORM=FIXUNB or =UNDEF). For other files of records, this entry must be included whenever records are undefined (RECFORM=UNDEF).

For fixed-length blocked disk or magnetic tape records, this entry specifies the num-

ber n of characters in an individual record. The input/output routines use this factor to block or deblock records, and to check record length of input records.

For undefined records, this entry specifies the number (r) of the general-purpose register that will contain the length of each individual input or output record. This may be any register 2-12. When undefined records are read, IOCS supplies the physical record size in the register. In the case of paper tape records, this will apply to both fixed unblocked and undefined records. When undefined records are built, the programmer must load the length of each record (in bytes) into the register before he issues the PUT instruction for the record. This becomes the count portion of the CCW that IOCS sets up for this file. Thus it determines the length of the record to be transferred to the output device. If an undefined punch or printer output record contains a control character in the main-storage output area (DTFSR CTLCHR specified), the length loaded into the RECSIZE register must also include one byte for this character.

REWIND=

If no specifications are given by the programmer, tape files are automatically rewound, but not unloaded, on an OPEN or CLOSE instruction and on an end-of-volume condition. If other operations are desired for a tape input or output file, this entry may be included with one of the following entered immediately after the = sign:

UNLOAD — to rewind the tape on OPEN, and to rewind and unload on CLOSE or an end-of-volume condition.

NORWD — to prevent rewinding the tape at any time.

TPMARK=NO

This entry is included if the user does not want a tapemark written as the first record on a tape output file if no labels are specified. This entry is also included if no tapemark is to be written following non-standard header labels. If this entry is omitted for a tape output file, a tapemark will be the first record if no labels are specified. Also if this entry is omitted, a tapemark will be written following non-standard header labels.

TRANS=Name

This entry applies to an input file read from the IBM 2671 Paper Tape Reader, and it specifies the symbolic name of a code-translation table. The table must conform to the specifications of the machine instruction TRANSLATE.

The input file records may be punched in 5-, 6-, 7-, or 8-channel paper tape, using any one of several different recording codes. If a code other than EBCDIC is used, it must be translated to EBCDIC code for use in System/360 programming. For IOCS to perform this translation, the user provides a translation table and specifies the symbolic name of the table in this TRANS entry. Then the logical IOCS routines translate the paper tape code and make the record available to the programmer in usable form directly in the input area.

TRUNCS=YES

This entry applies to disk files with fixed-length blocked records (RECFORM=FIXBLK) when short blocks are to be processed. It must be included:

- For an output file if the TRUNC macro instruction is to be issued in the problem program.
- For an input file if the TRUNC macro was issued to write short blocks when the file was originally created.

TYPEFLE=

This entry must be included to specify the type of file (input, output, or combined).

INPUT must be specified for:

2311 disk input files (with or without updating)

2400 magnetic tape input files

1442, 2501, 2520, 2540 card reader files

1052 keyboard input (both GET and PUT instructions may be issued)

1285 Optical Reader files

1287 Optical Reader files

OUTPUT must be specified for:

2311 disk output files

2400 magnetic tape output files

1442, 2520, 2540 card punch files

1403, 1404, 1443, 1445 printer output

1052 printer output (only PUT instructions may be issued)

CMBND must be specified for a 1442, 2520, or 2540 card file that is to be updated. That is, card records are to be read, processed, and then punched (PUT) in the same cards from which they were read. Thus input and output operations are combined for the same file. This operation can be performed in the IBM 1442 or the IBM 2520 or in the IBM 2540 if the punch-feed-read special feature is installed and cards are fed and read in the punch feed. (See PUT Macro: Updating.)

UPDATE=YES

This entry must be included if a disk input file (TYPEFLE=INPUT) is to be updated. That is, disk records are to be read, processed, and then transferred back (PUT) to the same disk record locations from which they were read.

VARBLD=(r)

Whenever variable-length blocked records are built directly in the output area (no work area specified), this entry must be included. It specifies the number (r) of a general-purpose register, which will always contain the length of the available space remaining in the output area. Any register 2-12 may be specified.

After the PUT instruction is issued for a variable-length record, IOCS calculates the space still available in the output area and supplies it to the programmer in this VARBLD register. The programmer then compares the length of his next variable-length record with the available space to determine if the record will fit in the area. This check must be made before the record is built. If the record will not fit, the programmer issues a TRUNC instruction to transfer the completed block of records to the tape file. Then the present record is built at the beginning of the output area, as the first record in the next block.

VERIFY=YES

This entry is included if the user wants disk records to be parity checked after they are written. If this entry is omitted, any records written on disk are not verified.

WLRERR=Name

This entry applies only to disk, magnetic tape, or paper tape input files. It specifies the symbolic name of a user's routine to which programming will branch if a wrong-length record is read. In his routine the user can perform any operation he desires for wrong-length records. However, he must not issue any GET macro instructions for this file. Also, if he uses any other IOCS macros in his routine, he must save the contents of Register 14. The address of the wrong-length record is supplied by IOCS in Register 1. At the end of his routine, the user must return to IOCS by branching to the address in Register 14.

Whenever fixed-length blocked records or variable-length records are specified (RECFORM=FIXBLK, =VARUNB, or =VARBLK), the machine check for wrong-length records is suppressed and IOCS generates a programmed check of record length. For fixed-length blocked records, record length is considered incorrect if the physical disk or tape record (gap to gap) that is read is not a multiple of the logical-record length (specified in DTFSR RECSIZE), up to the maximum length of the block (specified in DTFSR BLKSIZE). This permits the reading of short blocks of logical records, without a wrong-length-record indication.

For variable-length records, record length is considered incorrect if the length of the disk or tape record is not the same as the block length specified in the four byte block length field.

When fixed-length unblocked records are specified (RECFORM=FIXUNB), IOCS checks for a wrong-length-record indication that may have been set as the result of an I/O operation.

If this WLRERR entry is omitted from the set of DTFSR entries but a wrong-length record is detected by IOCS, one of the following will result:

- If the DTFSR ERROPT entry is included for this file, the wrong-length record will be treated as an error block and handled according to the user's specifications for an error (IGNORE, SKIP, or Name of error routine).
- If the DTFSR ERROPT entry is not included, the wrong length record is ignored.

The WLRERR entry does not apply to undefined records. Undefined records are not checked for incorrect record length.

WORKA=YES

Input/output records can be processed, or built, in work areas instead of the input/output areas. If this is planned, this entry WORKA=YES must be included, and the programmer must set up the work area(s) in main storage. The symbolic name used in the DS instruction that reserves the work area (or a general register containing the address of the work area) must be specified in each GET or PUT. On a GET or PUT, IOCS moves the record to, or from, the specified workarea.

Whenever this entry is included for a file, the DTF entry IOREG must be omitted.

The DTFEN Card

An end-of-definition card must follow the last set of DTFSR cards that applies to a magnetic tape file or to a DASD file if tape or DASD I/O modules are being assembled with the DTFs. (If tape or DASD I/O modules are assembled separately, this card need not be included.) It must be punched with DTFEN in the operation field and blanks in the name field. The operand field may be blank or it may contain OVLAY as a parameter (to provide compatibility with BOS). DOS uses the DTFEN card as a signal to begin generation of the required I/O modules.

DIRECT ACCESS METHOD (DTFDA)

The DTFDA detail entries that apply to a file when records are processed by the direct access method are explained in the following text and shown in DTFDA Entries (Figure 29).

Enter the symbolic name of the file in the name field and DTFDA in the operation field.

AFTER=YES

This entry must be included if any records (or an additional record) are to be written in a file by a format WRITE (count, key and data) following the last record previously written on a track. The remainder of the track is erased. That is, whenever the macro instruction WRITE Filename,AFTER or WRITE Filename,RZERO will be used in a program, this entry is required.

BLKSIZE=n

This entry indicates the size of the I/O area by specifying the maximum number, n, of characters that will be transferred to, or from, the area at any one time. When undefined records are read or written, the area must be large enough to accommodate the largest record.

If key length is specified by DTFDA KEYLEN, and if macro instructions that transfer the key areas of records will be issued, this area must provide for both the key area and data area of a record (see IOAREA1 and Figure 28. The length must specify an additional eight bytes if WRITE=AFTER is specified. If a file is to be created or if additional records are to be written in a file, the count area of the records must be included in this specification.

IOCS uses this specification to construct the count field of the CCW for reading or writing fixed-length records.

CONTROL=YES

This entry must be included if a CNTRL macro instruction will be issued for this file. The CNTRL macro for seeking on the 2311 allows the user to specify a track address to which access movement should begin for the next READ or WRITE instruction for a file. While the arm is moving, the programmer may process data and/or request I/O operations on other devices.

For the 2321, the CNTRL macro enables the user to seek to a specific address or to restore the strip to its subcell.

DEVICE={2311 / 2321}

This entry specifies whether the logical file is on a 2311 disk drive or a 2321 data cell drive. If this entry is omitted, 2311 is assumed.

ERRBYTE=Name

This entry is required for IOCS to supply indications of exceptional conditions to the problem program. The symbolic name of a 2-byte field, in which IOCS can store the error-condition or status codes, is entered after the = sign.

The codes are available for testing by the problem program at WAITF time after the transfer of a record has been completed One or more of the following Error status indication bits may be set to one by IOCS as in the bits indicated:

Byte	Bit	Error/Status Code Indication	Explanation
0	0	---	---
0	1	Wrong-length record	The wrong-length record indication is applicable for undefined records or fixed-length records. Fixed-length records: This bit is set on whenever the data length or key length of a record differs from the previous record. If an updated record is shorter than the original record, the updated record is padded with binary zeros to the length of the original record. If an updated record is longer than the original record, the original record positions are filled and the rest of the updated record is truncated and lost. Undefined-length records: This bit is set on under the following conditions: • When a READ is issued and the record is greater than the maximum data size (BLKSIZE minus KEYLEN, or BLKSIZE minus the value of KEYLEN plus eight if AFTER is used), a wrong-length error condition is given and the value returned in the RECSIZE register will be that of the actual record length. • When a WRITE ID or KEY is issued and the record to be written is greater than the maximum data size, a wrong-length error condition is given and the record written is equal to that of the maximum data length. If the DASD record is larger than the maximum data size, the remainder of the record is padded with binary zeros. The value in the RECSIZE register will be set equal to that of the maximum data length. • When a WRITE AFTER is issued and the record to be written is greater than the maximum data size, a wrong-length error condition is given and the record written is truncated to the maximum data length. The value in the RECSIZE register will be set equal to that of the maximum data length.
0	2	---	---
0	3	---	---
	4	No room found	The no-room-found indication is applicable only when the WRITE AFTER form of the macro is used for a file. The bit is set on if IOCS determines that there is not enough room left on the track to write the record. The record is not written.
0	5	---	---
0	6	---	---
0	7	---	---
1	0	Data check in count area	This is an unrecoverable error.

Byte	Bit	Error/Status Code Indication	Explanation
1	1	Track overrun	The number of bytes on the track exceeds the theoretical capacity. (Should not occur when DOS/360 macro instructions are used.)
1	2	End of cylinder	The end-of-cylinder indication bit is set on when SRCHM is specified for READ or WRITE KEY and the end-of-cylinder is reached before the record is found. If IDLOC is also specified, certain conditions also turn this bit on (for further information see IDLOC under DTFDA).
1	3	Data check when reading key or data	This is an unrecoverable error.
1	4	No record found	The no-record-found indication is given when a search ID or key (without SRCHM) is issued and a record is not found. If SRCHM=YES is specified, only the end of cylinder indication is set on.
1	5	End of file	The end-of-file indication is applicable only when the record to be read has a data length of zero. The ID returned in IDLOC, if specified, is hexadecimal FFFFF. The bit is set only after all the data records have been processed. For example, in a file having n data record (record n + 1 is the end-of-file record), the end-of-file indicator is set on when the user reads the n + 1 record.
1	6	End of volume	The end-of-volume indication is given in conjunction with the end-of-cylinder indication. This bit is set on if the next record ID (n + 1, 0, 1) that is returned on the end of the cylinde: is higher than the volume address limit. The volume address limit is cylinder 199, head 9, for a disk pack and subcell 19, strip 5, cylinder 4, head 19 for a data cell. These limits allow for the reserved alternate track area. If both the EOC and EOV indicators are set on, the ID returned in IDLOC is FFFFF.
1	7	---	---

IDLOC=Name

This entry is included if the programmer wants IOCS to supply the ID of a record after each READ or WRITE (ID or KEY) is completed. The symbolic name of a 5-byte field, in which IOCS is to store the ID, is specified after the = sign in this parameter. WAITF should be used before referencing this field.

IOCS supplies the ID of the record specified in the READ/WRITE instruction, or the ID of the next record location. The following may occur when this option is taken.

- Whenever a READ or WRITE ID (or if a READ or WRITE KEY without SRCHM) is issued, the address returned will be that of the next record location.
  Exception: When the record to be read or written is the last record of the cylinder, an end-of-cylinder indication is posted in ERRBYTE1, bit 2, and the address returned is that of the first record of the next cylinder. Also, the end-of-volume indication is posted, and the address returned in IDLOC will be all one bits. See Note.

- Whenever a READ or WRITE KEY with SRCHM is specified, the address returned will be that of the same record location.
  Exception: When the record is not found an end-of-cylinder condition is posted and the information returned is unpredictable.

For more information on the SRCHM specification see Figure 27.

Note: If IDLOC is specified and end-of-cylinder is reached on a 2311 file, the cylinder number will be increased by one, the head number will be set to zero, and the record number will be set to one. On a 2321 file, an end-of-cylinder condition with IDLOC specified will cause the high-order position of the head number to be increased by one, the low-order position of the head number to be set to zero, and the record number to be set to one. An overflow from the high-order position of the head number will cause the low-order position of the cylinder number to be increased by one, and the high-order position of the head number to be set to zero. The low-order position of the head number will be zero, and the record number will be set to one. Subsequent overflows of address locations will cause increases in the next higher positions of the addresses. (It is the user's responsibility to check the validity of the address returned in IDLOC.)

IOAREA1=Name

This entry must be included to specify the symbolic name of the input/output area used by this file. The input/output routines will transfer records to or from this area. The specified name must be the same as the name used in the DS instruction that reserves this area of main storage.

The main-storage input/output area must be large enough to contain the maximum number of bytes that will be required in any READ or WRITE instruction issued for this file in the problem program. This is affected by the length of record data areas, and by the use of the count and key areas as follows:

- If undefined records are specified in the DTFDA entry RECFORM, the area must provide space for the largest data record that will be processed.

- If the DTFDA entry KEYLEN is specified and if any instructions that read or write the key area of a record are to be issued in the problem program, the input/output area must provide room for the key area as well as the data area. The length needed for the key is the length specified in KEYLEN.

- If any write instructions that transfer the count area to a disk record will be issued in the problem program, eight bytes of main storage must be allotted at the beginning of the I/O area. In these eight bytes IOCS will construct the count field to be transferred to disk.

MACRO INSTRUCTION	ID SUPPLIED	
	With SRCHM	Without SRCHM
READ Filename,KEY	Same record	Next record
READ Filename,ID	Next Record	Next record
WRITE Filename,KEY	Same record	Next record
WRITE Filename,ID	Next Record	Next record
WRITE Filename,RZERO	None	None
WRITE Filename,AFTER[,EOF]	None	None

Figure 27. ID Supplied After a READ or WRITE Instruction

Whenever a WRITE instruction is issued, IOCS assumes that the input/output area (see Figure 7) contains the information implied by the type of instruction that is to be executed (Figure 28).

KEYARG=Name

This entry must be included if records are to be identified by key. That is, if the macro instruction READ Filename,KEY or WRITE Filename,KEY will be issued in the problem program, this entry is required. KEYARG specifies the symbolic name of the key field in which the user will supply th record key for the READ/WRITE routines.

MACRO INSTRUCTION	I/O AREA CONTENTS	
	With KEYLEN	Without KEYLEN
READ Filename,KEY	Data	
READ Filename,ID	Key and Data	Data
WRITE Filename,KEY	Data	
WRITE Filename,ID	Key and Data	Data
WRITE Filename,RZERO	Anything	Anything
WRITE Filename,AFTER[,EOF]	Count, Key, and Data	Count and Data

Figure 28. I/O Area Requirements for DA

OPERATION	OPERAND #	MUST BE INCLUDED	REMARKS *
DTFDA†		For each file	Header Card. Specify 7-character or less symbolic filename in the name field.
	AFTER = YES	If record reference AFTER used for an output record	CAPREC is assumed.
	†BLKSIZE = n	For each file	Length of I/O area. n = maximum number of characters.
	CONTROL = YES	For CNTRL macro used	
	†DEVICE = 2311 / 2321	For each 2321 file	Identifies DASD unit
	†ERRBYTE = Name	For each file	Symbolic name of 2-byte field for error/status codes supplied by IOCS.
	IDLOC = Name	If ID of same or next record to be supplied by IOCS	Symbolic name of 5-byte field for ID.
	†IOAREA1 = Name	For each file	Symbolic name of input/output area. Same as used in DS.
	KEYARG = Name	If record reference by key	Symbolic name of key field.
	KEYLEN = n	If records contain key areas	All keys must be the same length. n = length of keys.
	LABADDR = Name	To check/write additional labels	Symbolic name of user's label routine.
	MODNAME = Name	If a name other than the standard or a more inclusive module is referenced.	
	READID = YES	If record reference by ID used for an input record	
	READKEY = YES	If record reference by key used for an input record	
	RECFORM = FIXUNB	For fixed-length records	If this entry is omitted, IOCS assumes FIXUNB.
	UNDEF	For records not fixed-length, or Records added to a file and EOF record written	
	RECSIZE = (r)	For undefined records	r = number of a register 2-12
	†SEEKADR = Name	For each file	Symbolic name of track-reference field. Field is 8 bytes long.
	SEPASMB = YES	If the DTF is assembled separately	
	SRCHM = YES	To search multiple tracks	Applies to record reference by key.
	†TYPEFLE = INPUT	For each file	Read and check standard labels.
	OUTPUT		Write standard labels.
	VERIFY = YES	To check record written on 2311	Assumed for 2321
	WRITEID = YES	If record reference by ID used for an output record	
	WRITEKY = YES	If record reference by key used for an output record	
	XTNTXIT = Name	To process extent card information	Symbolic name of user's extent routine.

† Must be included. Other entries are included when applicable.

When two choices are shown, select only the appropriate one and enter it after the = sign.

* The header card and each detail card except the last one used in a file set must contain a continuation punch in column 72. Each detail card except the last one used must also contain a comma after the last operand.

In all entries: Solid caps must be entered as shown (For example, AFTER = YES)

Lowercase letters are to be replaced by programmer's symbolic name or a number (For example, Filename in header card, or BLKSIZE = n where n is replaced).

n is a decimal self-defining value.

Figure 29. DTFDA Entries

When record reference is by key, IOCS uses this specification at assembly time to construct the data address field of the CCW for search commands.

KEYLEN=n

This entry must be included if record reference is by key or if keys are to be read or written. It specifies the number, n, of bytes in each key. All keys must be the same length. If this card is omitted, IOCS assumes a key length of zero.

If there are keys recorded on DASD and this entry is absent, a WRITEID or READID will read or write the data portion of the record.

When record reference is by key, IOCS uses this specification to construct the count field of the CCW for this file. IOCS also uses this in conjunction with IOAREA1 to determine where the data field in the main-storage I/O area is located (see IOAREA1).

LABADDR=Name

The user may require one or more user labels in addition to the standard file label. If so, he must include his own routine to check, or write, the labels. The symbolic name of his routine is specified in this entry. IOCS branches to this routine after it has processed the standard label. See the sections Writing and Checking DASD User Standard Labels for a complete discussion of the function of the LABADDR routine.

MODNAME=Name

This entry may be used to specify the name of the logic module that will be used with the DTF table to process the file. If the logic module is assembled with the program, the MODNAME in the DTF macro instruction must specify the same name as the DAMOD macro instruction. If this entry is omitted, standard names will be generated for calling the logic module. If two DTF macro instructions call for different functions that can be handled by a single module, only one module will be called.

READID=YES

This entry must be included if any input records are to be specified by ID (identifier) in the problem program. That is, whenever the macro instruction READ Filename,ID will be used in the program, this entry is required.

READKEY=YES

This entry must be included if any input records are to be specified by key in the problem program. That is, whenever the macro instruction READ Filename,KEY will be used in the program, this entry is required

RECFORM=

This entry specifies the type of records in the input, or output, file. Either of the following specifications may be entered immediately after the = sign:

FIXUNB for fixed-length records. All records are considered unblocked in the DAM method. If the user wants blocked records, he must provide his own blocking and deblocking.

UNDEF for undefined records. This specification is required only if the records are not fixed length.

RECSIZE=(r)

This entry must be included if undefined records are specified (RECFORM=UNDEF). It specifies the number (r) of the general-purpose register that will contain the length of each individual input or output record. This may be any register 2-12.

Whenever each undefined record is read, IOCS supplies the length of the data area of that record in the register.

When an undefined record is to be written in the file, the programmer must load the length of the data area of the record (in bytes) into this register, before he issues the WRITE instruction for the record. IOCS adds the length of the key when required.

When records are to be written in the file (AFTER specified in the WRITE instruction), IOCS uses the length in constructing the count area to be written on DASD. IOCS adds the length of both the count and the key when required.

SEEKADR=Name

This entry must be included to specify the symbolic name of the user's track-reference field. In this field the user stores the track location of the particular record to be read or written. The READ, WRITE, and CNTRL routines refer to this field to determine which volume and which track on that volume contains the desired record. Whenever records are to be located by searching for a specified ID, the track-reference field must also contain the number of the record on the track.

The track-reference field is an eight-byte field (MBBCCHHR), and the symbolic name labels the first byte.

The bytes are used for:

M Symbolic unit number (0-254)

BB Always 00 for 2311. For 2321, the first byte is always 0. The second byte specifies cell number (0-9) for 2321.

CC For 2311 the first byte is always 0. The second byte contains the cylinder number (0-199) in which the record is located. For the 2321, the first byte contains the number of the sub-cell (0-19). The second byte contains the number of the strip (0-9).

HH For 2311 the first byte is always 0. The second byte contains the number of the read/write head that applies to the record. For 2321 the first byte specifies one of the five head bar positions (0-4). The second byte specifies one of the twenty head elements (0-19).

R Sequential number of the record on the track (0-255).

SEPASMB=YES

Include this parameter if the DTF is assembled separately. This causes a CATALR card with the filename to be punched ahead of the object deck and defines the Filename as an ENTRY point in the assembly.

SRCHM=YES

If input/output records will be identified by key, this entry may be included to cause IOCS to search multiple tracks for each specified record. The instruction READ Filename,KEY or WRITE Filename,KEY will cause a search of the track specified in the track-reference field and all following tracks in the cylinder, until the record is found or the end of the cylinder is reached. If the logical file ends before the end of the cylinder and the record is not found, the search continues into the next file, if any, on the cylinder. EOC, instead of NRF, will be indicated.

Without this card, each search is confined to the specified track.

TYPEFLE=

This entry must be included to indicate how standard volume and file labels are to be processed:

INPUT Standard labels are to be read and checked.

OUTPUT Standard labels are to be written.

Because logical files on DASD must always contain labels, this entry is always required.

VERIFY=YES

This entry is included if the user wants to check the parity of 2311 records after they are written. VERIFY is always assumed when 2321 records are written. If this card is omitted, any records written on 2311 are not verified.

WRITEID=YES

This entry must be included if the DASD storage location for writing any output record is to be specified by record ID (identifier) in the problem program. That is,

whenever the macro instruction WRITE Filename,ID will be used in the program this card is required.

WRITEKY=YES

This entry must be included if the DASD location for writing any output record is to be specified by record key in the problem program. That is, whenever the macro instruction WRITE Filename,KEY will be used in the program, this card is required.

XTNXIT=Name

This entry is included if the programmer wants to process XTENT card information. It specifies the symbolic name of the user's XTNTXIT routine. During an OPEN, IOCS branches to the user's routine after each specified extent has been checked and validated. Upon entering the user's routine, IOCS stores in register 1 the address of a 14-byte field that contains the extent card information (in binary form).

BYTES	CONTENTS
0	Extent type code (as specified in the XTENT card)
1	Extent sequence number
2-5	Lower limit of the extent (CCHH)
6-9	Upper limit of the extent (CCHH)
10-11	Symbolic unit
12	Old bin number
13	Present bin number of the extent (B2)

The user returns to IOCS by use of the LBRET macro instruction.

DIRECT ACCESS MODULE (DAMOD) PARAMETERS

A set of DAMOD entries is included for each DAM logic module necessary to support each DTFDA macro in a particular problem program. The logic modules are described by a DAMOD header entry and a series of keyword parameters.

The header entry contains DAMOD in the operation field and may contain a user module name in the name field. The parameters are explained in the following text and shown in Figure 30.

Name	Operation	Operand	Remarks
[Modname]	DAMOD		Must be included.
		AFTER=YES	When WRITE with the operand AFTER or RZERO is used.
		IDLOC=YES	Required if IDLOC specified in DTFDA.
		RECFORM=FIXUNB / UNDEF	Describes record format.
		SEPASMB=YES	If the module is assembled separately.

Figure 30. DAMOD Entries

AFTER=YES — This entry causes generation of a logic module that can format write (count, key, and data). It perform the functions required by WRITE Filename,AFTER and WRITE Filename,RZERO. The module will also process any files in which the AFTER parameter is not specified in the DTF.

IDLOC=YES — This entry causes generation of a logic module whic will handle the return of record identifier (ID) information to the user. The module will also process any files in which the IDLOC parameter is not specified in the DTF.

RECFORM={UNDEF / FIXUNB} — If UNDEF is specified, the logic module that is generated can handle both unblocked fixed length and undefined records. If the entry is omitted, or if FIXUNB is specified, the logic module that is generated can handle only fixed-length unblocked records.

SEPASMB=YES — Include this parameter if the logic module is assembled separately. This causes a CATALR card with the module name (standard or user) to be punched ahead of the object deck.

Recommended Module Name List for DAMOD

Each name will begin with a 3-character prefix (IJI) and consist of a 5-character field corresponding to the options permitted in the generation of the module.

DAMOD name = IJIabcde

a = F if RECFORM=FIXUNB

= B if RECFORM=UNDEF (handles both UNDEF and FIXUNB)

b = A if AFTER=YES is specified

= Z if AFTER is not specified

c = I if IDLOC is specified

= Z if IDLOC is not specified

d = Z always

e = Z always

Subsetting and Supersetting of DAMOD Names

The following diagram illustrates the subsetting and supersetting allowed for DAMOD names. Three parameters allow supersetting. For example, the module IJIBAIZZ is a superset of the module with the name IJIFAZZZ.

```
      +++
  IJI BAIZZ
      FZZ

+Subsetting/supersetting permitted.
```

INDEXED SEQUENTIAL SYSTEM (DTFIS)

The DTFIS detail entries that apply to a file when records are processed by the Indexed Sequential Management System are explained in the following text and summarized in Figure 31. A DTFIS header entry and a series of detail entries describe the file.

The symbolic name of the file, Filename, is entered in the name field. DTFIS is entered in the operation field.

Status or Condition Code Indication

The DTF macro instruction provides a two-byte field where all status or condition codes will be placed after execution of each macro instruction. (Only the first byte is used; the second byte is reserved for future use.) This field can be referenced by the user as FilenameC. Filename should be the same as that specified in the DTF header entry for the file. The FilenameC byte will be in the format shown in Figure 32.

The user has the facility of addressing certain fields or locations in a table, such as FilenameC. ISFMS provides addressability for FilenameC by returning the address of the DTF table in register 1 after each ISFMS imperative macro is executed.

CYLOFL=n

This entry must be included if cylinder overflow areas are to be reserved for a logical file.

To reserve the areas for cylinder overflow this entry is required when a file is to be loaded onto DASD and when records are to be added to an organized file. It specifies the number n of tracks to be reserved on each cylinder.

If an independent overflow area is specified (by an XTENT card) along with the CYLOFL entry, overflow records are written in the independent overflow area after a cylinder overflow area becomes filled.

DEVICE={2311 / 2321}

This entry specifies the unit that contains the prime data area or overflow areas for the logical file.

DSKXTNT=n

This entry must be included to specify the maximum number n of extents for this file. The number must include all the data area extents if more than one DASD area is used for the data records, and all the index area and independent overflow area extents that are specified by XTENT cards. Thus the minimum number specified by this entry is 2: one extent for one prime data area, and one for a cylinder index.

Each area assigned to an ISFMS data file is considered an extent.

Note: Master and cylinder indices are treated as one area.

HINDEX={2311 / 2321}

This entry specifies the unit containing the highest index.

IOAREAL=Name

This entry must be included when a file is created (loaded) or when records are added

to an organized file. It specifies the symbolic name of the output area used for loading or adding records to the file. The specified name must be the same as the name used in the DS instruction that reserves this area of main storage. The ISFMS routines construct the contents of this area and transfer records from this area to DASD.

This main-storage output area must be large enough to contain the count area, key area, and data area of records. Furthermore, the data-area portion must provide enough space for the sequence-link field of overflow records whenever records are added to a file (Figure 33).

OPERATION	OPERAND#	MUST BE INCLUDED	REMARKS*
DTFIS†		For each file	Header card. Specify 7-character or less symbolic filename in name field.
	CYLOFL=n	Cylinder overflow areas	May be specified alone or with an independent overflow area. n = number of tracks for each cylinder.
	†DSKXTNT=n	For each file	Maximum number of extents specified for the file.
	DEVICE=2311/2321	For each 2321 file	Specifies unit containing the prime data area.
	HINDEX=2311/2321		Specifies Unit containing high level indices.
	IOAREAL=Name	IOROUT specifies LOAD, ADD, or ADDRTR	Symbolic name of input/output area. Same as used in DS. At least one I/O area must be specified for a file.
	IOAREAR=Name	TYPEFLE specifies RANDOM or RANSEQ	
	IOAREAS=Name	TYPEFLE specifies SEQNTL or RANSEQ	
	IOREG=(r)	Process blocked records in I/O area	r = number of register 2-12
	†IOROUT=LOAD	For each file	Build or extend a file on DASD.
	ADD		Insert new records in an organized file.
	RETRVE		Retrieve records for processing/updating.
	ADDRTR		Insert and retrieve records.
	KEYARG=Name	Required if random retrieval or sequential retrieval is initiated by key.	Symbolic name of key field in main storage.
	†KEYLEN=n	For each file	All keys must be the same length. n = length of key.
	KEYLOC=n	With blocked records.	n = high-order position of key field within each record.
	MODNAME=Name	If a name other than the standard or a more inclusive module is referenced.	
	MSTIND=YES	For Master Index	
	NRECDS=n		n = number of records in a block. If RECFORM=FIXUNB, n is assumed to be 1.
	†RECFORM=FIXUNB	For Unblocked records	Applies to records in prime data area only.
	FIXBLK	For Blocked records	
	†RECSIZE=n	For each file	n = number of characters in each logical record.
	SEPASMB=YES	If the DTF is assembled separately.	
	TYPEFLE=RANDOM	IOROUT specified RETRVE or ADDRTR	Random processing.
	SEQNTL		Sequential processing.
	RANSEQ		Random and sequential processing.

Figure 31. DTFIS Entries (Part 1 of 2)

OPERATION	OPERAND#	MUST BE INCLUDED	REMARKS*
DTFIS †	VERIFY=YES	To check records written on 2311	Assumed for 2321
	WORKL=Name	If IOROUT specifies LOAD, ADD, or ADDRTR	
	WORKR=Name	If TYPEFLE specifies RANDOM or RANSEQ and records are processed in a work area	Takes precedence over IOREG.
	WORKS=YES	If TYPEFLE specifies SEQNTL or RANSEQ and records are processed in work areas	Takes precedence over IOREG.

†Must be included. Other entries are included when applicable.

#When two choices are shown, select only the appropriate one and enter it after the sign.

*The header card and each detail card except the last one used in a file set must contain a continuation punch in column 72. Each detail card except the last one used must also contain a comma after the last operand.

In all entries: Solid caps must be entered as shown (For example, IOROUT=LOAD).

Lowercase letters are to be replaced by programmer's symbolic name or a number (For example, Filename in header card, or CYLOFL=n where n is replaced).

n is a decimal self-defining value.

Figure 31. DTFIS Entries (Part 2 of 2)

ADD, RETRVE, and ADDRTR

Bit	Cause	Explanation
0	DASD error	Any uncorrectable DASD error has occurred (except wrong length record).
1	Wrong length record	A wrong length record has been detected during an I/O operation.
2	End of file	The EOF condition has been encountered during execution of the sequential retrieval function.
3	No record found	The record to be retrieved has not been found in the data file. This applies to Random (RANSEQ) and to SETL in SEQNTL (RANSEQ) when KEY is specified.
4	Illegal ID specified	The ID specified to the SETL in SEQNTL (RANSEQ) is outside the prime data file limits.
5	Duplicate record	The record to be added to the file has a duplicate record key of another record in the file.
6	Overflow area full	An overflow area in a cylinder is full, and no independent overflow area has been specified, or an independent overflow area is full, and the addition cannot be made. The user should assign an independent overflow area or extend the limit.
7	Overflow	The record being processed in one of the retrieval functions (RANDOM/SEQNTL) is an overflow record.

LOAD

Bit	Cause	Explanation
0	DASD error	Any uncorrectable DASD error has occurred (except wrong length record).
1	Wrong length record	A wrong length record has been detected during an I/O operation.
2	Prime data area full	The prime data area has been filled while creating or extending the data file. The user should extend the upper limit by use of a new extent card.
3	Cylinder Index area full	The Cylinder Index area is not large enough to contain all the entries needed to index each cylinder specified for the prime data area. This condition can occur during the execution of the SETFL. The user must extend the upper limit of the cylinder index by using a new extent card.
4	Master Index full	The Master Index area is not large enough to contain all the entries needed to index each track of the Cylinder Index. This condition can occur during SETFL. The user must extend the upper limit, if he is creating the file, by using an extent card. Or, he must reorganize the data file and assign a larger area.
5	Duplicate record	The record being loaded is a duplicate of the previous record.
6	Sequence check	The record being loaded is not in the sequential order required for loading.

Figure 32. FilenameC--Status or Condition Code Byte

IOAREAR=Name

This entry must be included whenever records are processed in random order. It specifies the symbolic name of the input/output area used for random retrieval (and updating). The specified name must be the same as the name used in the DS instruction that reserves this area of main storage.

This main-storage I/O area must be large enough to contain the data area of records. Furthermore, the data-area portion must provide enough space for the sequence-link field of overflow records (Figure 34).

IOAREAS=Name

This entry must be included whenever records are processed in sequential order by key. It specifies the symbolic name of the input/output area used for sequential retrieval (and updating). The specified name must be the same as the name used in the DS instruction that reserves this area of main storage.

This main-storage I/O area must be large enough to contain the key area and data area of records for unblocked data records and the data area for blocked records. Furthermore, the data-area portion must provide enough space for the sequence-link field of overflow records (Figure 34).

IOREG=(r)

This entry must be included whenever records are to be retrieved and processed directly in the I/O area. It specifies the number (r) of the register that ISFMS can use to indicate which individual record is available for processing. ISFMS puts the address of the current record in this register each time a READ, WRITE, GET, or PUT is executed. Any register 2-12 may be specified.

IOROUT=

This entry must be included to specify the type of function to be performed. One of the following specifications is entered after the = sign:

LOAD To build a logical file on DASD or to extend a file beyond the highest record presently in an organized file.

ADD To insert new records into an organized file.

RETRVE To retrieve records from a file for either random or sequential processing and/or updating.

ADDRTR To both insert new records into a file (ADD) and retrieve records for processing and/or updating (RTR).

KEYARG=Name

This entry must be included for random READ/WRITE operations and sequential retrieval initiated by key. It specifies the symbolic name of the main-storage key field in which the user must supply the record key to ISFMS.

FUNCTION	OUTPUT AREA REQUIREMENTS (IN BYTES)			
	Count	Key	Sequence Link	Data
Load Unblocked Records	8	Key Length	—	Record Length
Load Blocked Records	8	Key Length	—	Record Length x Blocking Factor
Add Unblocked Records	8	Key Length	10	Record Length
Add Blocked Records	8	Key Length	— OR*	Record Length x Blocking Factor
	8	Key Length	10	Record Length
* Whichever Is Larger				

Figure 33. Output Area Requirements for Loading or Adding Records to a File by ISFMS

FUNCTION	I/O AREA REQUIREMENTS (IN BYTES)			
	Count	Key	Sequence Link	Data
Retrieve Unblocked Records	—	Key Length for sequential unblocked records	10	Record Length
Retrieve Blocked Records	—		— OR*	Record Length x Blocking Factor
	—		10	Record Length
* Whichever Is Larger				

'igure 34. I/O Area Requirements for Random or Sequential Retrieval by ISFMS

:EYLEN=n

'his entry must be included to specify the ıumber, n, of bytes in the record key. All eys must be the same length.

:EYLOC=n

'his entry must be included if an add, oad, or retrieve function is to be per- ormed and blocked records are specified n DTFIS RECFORM. This entry must always e included for blocked records. It upplies ISFMS with the high-order position f the key field within the data record. hat is, if the key is recorded in posi- ions 21-25 of each record in the file, this ntry specifies 21.

ISFMS uses this specification to locate by key) a specified record within a block. he key area of a DASD record contains the ey of the highest record in the block. To earch for any other records, ISFMS locates he proper block and then examines the key ield within each record in the block.

:ODNAME=Name

his entry may be used to specify the name f the logic module that will be used with he DTF table to process the file. If the ogic module is assembled with the program, he MODNAME in the DTF macro instruction ıust specify the same name as the ISMOD ıacro instruction. If this entry is omit- ed, standard names will be generated for alling the logic module. If two DTF macro nstructions call for different functions hat can be handled by a single module, only ne module will be called.

MSTIND=YES

This entry is included whenever a master index is used for a file. In this case, it is required when a file is loaded (to instruct ISFMS to build the index) and when records are added to or retrieved from a file with a master index.

ISFMS always builds a track index and a cylinder index, but the master index is optional. The master index, if used, is the highest level index, and it includes an index record for each track of the cylinder index. Thus, it points to the cylinder index on a search for a particular record (see Indices: Master Index). The location of the master index is specified by a Job Control XTENT card.

NRECDS=n

This entry specifies the number, n, of logical records in a block (called the blocking factor). If RECFORM=FIXUNB, n is assumed to be 1.

RECFORM=

This entry specifies the type of records in the logical file. All logical records in the file must be fixed length. However, they may be either blocked or unblocked. One or the other of these specifications must be entered after the = sign:

FIXUNB for unblocked records.

FIXBLK for blocked records. With this specification the key of the highest record in the block becomes the key for the block and must be recorded in the key area.

The specification that is included when the logical file is loaded into DASD storage must also be included whenever the file is processed.

Records in the overflow area(s) are always unblocked (see Addition of Records and Overflow Areas), but that does not affect this entry. RECFORM refers to records in the prime data area only.

RECSIZE=n

This entry must be included to specify the number n of characters in a logical record. This is the length of the data area of each individual record. All logical records must be the same size.

SEPASMB=YES

Include this parameter if the DTF is assembled separately. This causes a CATALR card with the filename to be punched ahead of the object deck and defines the Filename as an ENTRY point in the assembly.

TYPEFLE=

This entry must be included when a retrieval function is to be performed. It specifies the type(s) of processing that is to be performed by the problem program for this file. One of the following specifications is entered after the = sign:

RANDOM	for random processing. Records are retrieved from the file in random order specified by key. Only READ instructions may be issued to transfer records.
SEQNTL	for sequential processing. The problem program specifies the first record to be retrieved, and thereafter ISFMS retrieves records in sequential order by key. The first record is specified by key, ID, or the beginning of the logical file (see SETL Macro). Only GET instructions may be issued to transfer records.
RANSEQ	for both random and sequential processing. Only READ and GET instructions may be issued to transfer records.

TYPEFLE is not required for loading or adding functions.

VERIFY=YES

This entry is included if the user wants to check the parity of 2311 records after they are written. If this entry is omitted, any records written on 2311 are not verified VERIFY is always assumed when 2321 records are written.

WORKL=Name

This entry must be included whenever a file is to be created (loaded) or records are to be added to an organized file. It specifies the symbolic name of the work area in which the user must supply the data records to ISFMS for loading or adding to the file. The specified name must be the same as the name used in the DS instruction that reserves this area of main storage.

This work area must provide space for one logical data record when a file is to be created (for blocked records--Data; for unblocked records--Key and Data).

Due to record shifting in the ADD function, the original contents of WORKL will be changed.

WORKR=Name

When records are processed in random order, this entry must be included if the individual records are to be processed in a work area rather than the I/O area. It specifies the symbolic name of the work area. This name must be the same as the name used in the DS instruction that reserves this area of main storage. This area must provide space for one logical record (data area).

When this entry is included and a READ or WRITE instruction is executed, ISFMS moves the individual record to, or from, this area.

WORKS=YES

When records are processed in sequential order, this entry must be included if the individual records are to be processed in work areas rather than the I/O area. Each GET and PUT instruction must specify the symbolic name of the work area to or from which ISFMS is to move the record. The

area must be large enough for one logical record (data area) and the record key (key area) when unblocked records are processed, and for one logical record (data area) when blocked records are processed.

The ISFMS workarea requirements are as follows:

	Unblocked Records	Blocked Records
Load	KL + DL	DL
Add	KL + DL or 10*	DL or 10*
Random Retrieve	DL	DL
Sequential Retrieve	KL + DL	DL

Where:

K=Key, D=Data, L=Length
*Whichever is greater.

INDEXED SEQUENTIAL MODULE (ISMOD) PARAMETERS

A set of ISMOD parameters is included for each logic module necessary to support each DTFIS macro in a particular problem program. The logic modules are described by a ISMOD header entry and a series of parameter entries. The header entry contains ISMOD in the operation field, and may contain a user supplied name in the name field. The parameters are explained in the following text and shown in Figure 35.

Name	Operation	Operand	Remarks
[Modname]	ISMOD		
		IOROUT= LOAD ADD RETRVE ADDRTR	Specifies function to be performed.
		RECFORM= FIXUNB FIXBLK BOTH	Describes file. Required if IOROUT specifies ADD or ADDRTR. If IOROUT specifies LOAD or RETRVE, BOTH is assumed.
		SEPASMB= YES	If the module is assembled separately.
		TYPEFLE= RANDOM SEQNTL RANSEQ	Required if IOROUT specifies RETRVE or ADDRTR.

Figure 35. ISMOD Entries

IOROUT={LOAD, ADD, RETRVE, ADDRTR}

This entry specifies the type of logic module required to perform a given function. The entries are listed in the following text.

LOAD generates a logic module for creating a file.

ADD generates a logic module for adding new records into an existent file.

RETRVE generates a logic module to retrieve (randomly/sequentially) records from an organized file.

ADDRTR generates a logic module that combines the features of the ADD and RETRVE modules. This module will also process any file in which only ADD or RETRVE is specified in the IOROUT parameter statement of the DTF for that file, and in which the TYPEFLE entry contains the corresponding parameter (or a subset of it).

RECFORM={FIXUNB, FIXBLK, BOTH}

This entry generates a detailed logic module that will create, add to, or process an unblocked (FIXUNB) or blocked (FIXBLK) data file. If BOTH is specified, a module is generated to process both unblocked and blocked files, and the DTF entry for the file may specify either FIXUNB or FIXBLK in the RECFORM parameter statement. The RECFORM entry is required only when IOROUT specifies ADD or ADDRTR. If IOROUT specifies LOAD or RETRVE, a module that handles fixed-length blocked and unblocked files is generated, and the entry is not required.

SEPASMB=YES Include this parameter if the logic module is assembled separately. This causes a CATALR card with the module name (standard or user) to be punched ahead of the object deck.

TYPEFLE={RANDOM / SEQNTL / RANSEQ}

This entry is required when IOROUT specifies RETRVE or ADDRTR. RANDOM generates a logic module that will include only random retrieval capabilities. SEQNTL generates a logic module that includes only sequential retrieval capabilities. RANSEQ generates a logic module that includes random and sequential capabilities. It will also process any file in which the TYPEFLE parameter statement of the DTF specifies either RANDOM or SEQNTL.

Recommended Module Name List for ISMOD

Each name will begin with a 3-character prefix (IJH) and will consist of a 5-character field corresponding to the options permitted in the generation of the module.

ISMOD name = IJHabcde

a = U if RECFORM=FIXUNB and IOROUT specifie ADD or ADDRTR
  = B if RECFORM=FIXBLK and IOROUT specifie ADD or ADDRTR
  = A if RECFORM=BOTH and IOROUT specifies ADD or ADDRTR
  = Z if RECFORM=BOTH and IOROUT specifies LOAD or RETRVE

b = L if IOROUT=LOAD
  = I if IOROUT=ADD
  = R if IOROUT=RETRVE
  = A if IOROUT=ADDRTR

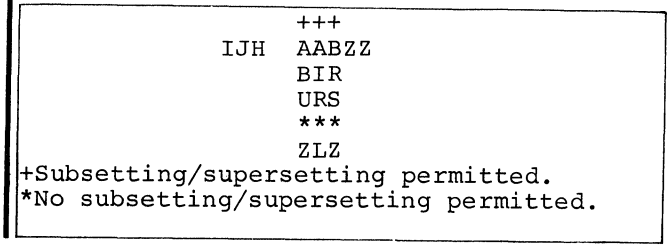
c = R if TYPEFLE=RANDOM is specified
  = S if TYPEFLE=SEQNTL is specified
  = B if TYPEFLE=RANSEQ is specified
  = Z if TYPEFLE is not specified

d = Z always

e = Z always

Subsetting and Supersetting of ISMOD Names

The following diagram illustrates the subsetting and supersetting allowed for ISMOD names. Three parameters allow supersetting For example, the module IJHAIBZZ is a superset of the module IJHURSZZ.



Records can be transferred to or from an input/output device by issuing physical IOCS macro instructions. These instructions relate directly to the physical IOCS routines and are distinct from logical IOCS routines. See the introductory section Physical IOCS vs Logical IOCS.

The user must provide any of the functions that are required for a problem program such as blocking or deblocking records, performing programmed wrong-length record checks, testing (the CCB) for certain errors, switching I/O areas when two areas are used, and setting up Channel Command Words (CCW). He must also recognize and bypass checkpoint records if they are interspersed with data records on an input tape.

Physical IOCS routines control the transfer of data to or from the external device. These routines are:

- Start I/O
- I/O Interrupt
- Channel Scheduler
- Device Error

Thus, physical IOCS macro instructions provide the user with the capability of obtaining data and performing nondata operations in I/O devices, with exactly the CCW's that he requests. For example, if he is handling only physical records, he does not need the logical IOCS routines for blocking and deblocking logical records.

Three macro instructions are available to the programmer for direct communication with physical IOCS: CCB (Command Control Block), EXCP (Execute Channel Program), and WAIT. These are explained in the following sections. Whenever physical IOCS macro instructions are used, the programmer must construct the Channel Command Words (CCW) for his input/output operations. He uses the assembler-instruction CCW statement to do this.

Macros normally used with files that are processed by logical IOCS are necessary when standard DASD or magnetic tape labels are to be processed, or when DASD file protect is present. The DTFPH, OPEN, CLOSE, LBRET, and FEOV macros can be used in this processing. See the discussion DTFPH (Define the File for Physical IOCS).

The OPEN and the DTFPH macros are also necessary when a 2311 is used as a checkpoint file.

CCB MACRO

Name	Operation	Operand
Blockname	CCB	SYSnnn,Command-list-name [,X'nnnn'][,Sense Address]

A CCB (Command Control Block) macro instruction must be specified in the problem program for each I/O device that is to be controlled by physical IOCS macro instructions. This block (see Figure 37) is necessary to communicate information to physical IOCS so that it can perform desired operations (for example, notify the problem program of printer channel 9). The Command Control Block also receives status information after an operation and makes this available for use by the problem program. The user should ensure proper boundary alignment of the CCB if necessary for his program.

Blockname: The CCB instruction must be labeled (Blockname) with a symbolic name. This name can be used as the operand in the EXCP and WAIT macro instructions that refer to the Command Control Block.

SYSnnn: Two operands are required in this CCB macro instruction. The first operand specifies the symbolic unit (SYSnnn) for the actual I/O unit with which this control block will be associated. The name may be SYSRDR, SYSLST, SYSIPT, SYSLOG, SYSPCH, SYSRES, SYS000-SYS244. The actual I/O unit can be assigned to the symbolic unit by a Job Control ASSGN statement.

Command-list-name: The second operand (Command-list-name) specifies the symbolic name of the first CCW to be used with this CCB. This name must be the same as the name specified in the assembler CCW statement that constructs the Channel Command Word.

X'nnnn': A hexadecimal value sets the CCB user option bits. Column 5 of Figure 36 gives the value used to set a user option bit ON. If more than one bit needs to be set, the sum of the values is used. For

Byte	Bit	Condition Indicated		On Values for Third Operand in CCB Macro	Mask for Test Under Mask Instruction
		1 (ON)	1 (OFF)		
2	0 Traffic Bit (WAIT)	I/O Completed. Normally set at Channel End. Set at Device End if Bit 5 is ON.	I/O Requested and not completed.		X'80'
	1 End of File on System Input	/* or /& on SYSRDR or SYSIPT. Byte 4, Unit Exception Bit is also ON.			X'40'
	2 Unrecoverable I/O Error	I/O error passed back due to program option or operator option	No program or operator option error was passed back	X'20'	
	3 * Accept Unrecoverable I/O Error (Bit 2 is ON)	Return to User after Physical IOCS Attempts to correct I/O Error. +	Operator Option: Dependent on the Error	X'1000'	X'10'
	4 * 2671 Data Check	Operator Options: Ignore, Retry, or Cancel	Operator Option: Retry or Cancel	X'0800'	X'08'
	Return any DASD Data Checks	Return to User			
	5 * Post at Device End	Device End Condition will be posted i.e., byte 2, bit 0 and byte 3, bits 3 and 6 set at Device End. Also byte 4, bit 5 is set (see Figure 36).	Device End Conditions will not be posted. Traffic Bit is set at Channel End.	X'0400'	X'04'
	6 * Return: Uncorrectable tape read data check; 2540 or 2520 punch equipment check; or DASD read or verify data check. (Data checks on count not returned.)	Return to user after physical IOCS attempts to correct tape or DASD error.	Operator Option: Ignore or Cancel for Tapes and punches. Retry or cancel for DASD.	X'0200'	X'02'
	7 * User Error Routine	User will handle error recovery (Test Bit 2). ¤	A Physical IOCS Error routine will be used.	X'0100'	X'01'
3	0 Data check in DASD count Field.	Yes - Byte 3, bit 3 is OFF; Byte 2, bit 2 is ON.	No		X'80'
	Data Check - 1285 or 1287 in Journal Tape Mode	Yes	No		
	1 DASD Track overrun.	Yes	No		X'40'
	Keyboard correction 1285 or 1287 in Journal Tape Mode	Yes	No		
	2 End of DASD Cylinder	Yes	No		X'20'
	Hopper Empty 1287 Document Mode	Yes	No		
	3 Tape read data check; 2540 or 2520 punch equipment check; or any DASD data check	Operation was unsuccessful. Byte 2, Bit 2 is also ON. Byte 3, Bit 0 is OFF	No		X'10'
	1285 or 1287 equipment check	Yes	No		

Figure 36. Conditions Indicated by CCB Bytes 2 and 3 (Part 1 of 2)

Byte	Bit	Condition Indicated		On Values for Third Operand in CCB Macro	Mask for Test Under Mask Instruction
		1 (ON)	0 (OFF)		
3	4 Questionable Condition Non-recovery	Card: Unusual Command sequence (2540). DASD: No record found. 1285/1287: Document Jam or Torn Tape			X'08'
	5 Unused: Must be zero				
	6 Verify Error for DASD or Carriage Channel 9 Overflow 1287 Document Mode - Late Stacker Select	Yes. (Set ON when Channel 9 is reached only if Byte 2, Bit 5 is ON.)	No		X'02'
	7* Command Chain Retry	Retry begins at last CCW executed.	Retry begins at first CCW of channel program.	X'0001'	X'01'

* User Option Bits. Set in CCB macro. Physical IOCS sets the other bits OFF at EXCP time and ON when the condition specified above occurs.
\+ I/O program check, command reject, or tape equipment check will always terminate the program.
¤ User must handle all error or exceptional conditions except Channel Control Check, Interfact Control Check, I/O Proaram Check, and I/O Protection Check.

Figure 36. Conditions Indicated by CCB Bytes 2 and 3 (Part 2 of 2)

example, to set user option bits 3, 5, 6 and 7 of byte 2 ON, X'1700' is used. (X'1700'=X'1000' + X'0400' + X'0200' + X'0100')

Sense Address: This operand, when supplied, causes a CCW for reading sense information to be generated as the last field of the CCB and sets the user error recovery bit (bit X'01' of byte 2) on. The name field (Sense Address) of the area that the user supplies must have a length attribute assigned of at least one byte. Physical IOCS uses this length attribute in the CCW to determine the number of bytes of sense information the user desires at his sense address.

Note: When user error recovery (CCB bit X'01' byte 2) is specified, the occurrence of a channel data check, unit check, or chaining check will cause byte 2, bit X'20' of the CCB to be turned on and completion posting and dequeuing to occur. I/O program and protection checks always cause program termination. Incorrect length and unit exception are treated as normal conditions (posted with completion). Note also that the user must request device end posting (CCB byte 2, bit X'04') in order to obtain errors after channel end.

CCB Format

From the specifications in this CCB instruction, the macro sets up a 16-byte or 24-byte field (Figure 37) as follows:

Bytes	Contents
0-1	After a record has been transferred, IOCS places the residual count from the CSW in these two bytes. By subtracting the residual count from the original count in the CCW, the problem program can determine the length of the record that was transferred.
2-3	The next two bytes are used for transmission of information between physical IOCS and the problem program. The problem program can test any bit in bytes 2 and 3, using the mask given in column 6 of Figure 36. More than one bit can be tested by using the hexadecimal sum of the test values. All bits are set at 0 (OFF) when the problem program is assembled unless the third parameter is

Bytes	Contents
	specified. If the third parameter is specified, it will be assembled into these two bytes. The user may turn on bit 7 in byte 3 and bits 3, 4, 5, 6, 7 in byte 2. During execution each bit may be set at 1 (ON), by the problem program or by a condition detected by physical IOCS. Any bits that can be turned on by physical IOCS, during program execution, are reset to zero by PIOCS the next time an EXCP macro using the same CCB is executed. The condition indicated by the setting of each bit is shown in Figure 36.
4-5	These two bytes are the status bytes of the CSW. If Device End posting is requested (byte 2, bit 5), Device End status will be OR'ed in. Byte 4 is set to X'00' at EXCP time.
6-7	These two bytes are a hexadecimal representation of the symbolic unit for the I/O devices, as specified in the first operand of this CCB.
8	This byte must contain X'00'.
9-11	These three bytes contain the address of the CCW (or first address of a chain of CCW's) associated with this CCB and specified symbolically in the second operand.
12	This byte must contain X'00'.
13-15	These bytes contain the address of the CCW in the CSW stored at channel end interrupt for this I/O operation.
16-23	These bytes are allotted only wher the Sense Address operand is supplied in the CCB Macro. They contain the CCW for returning sense information to the problem prograr

EXCP MACRO

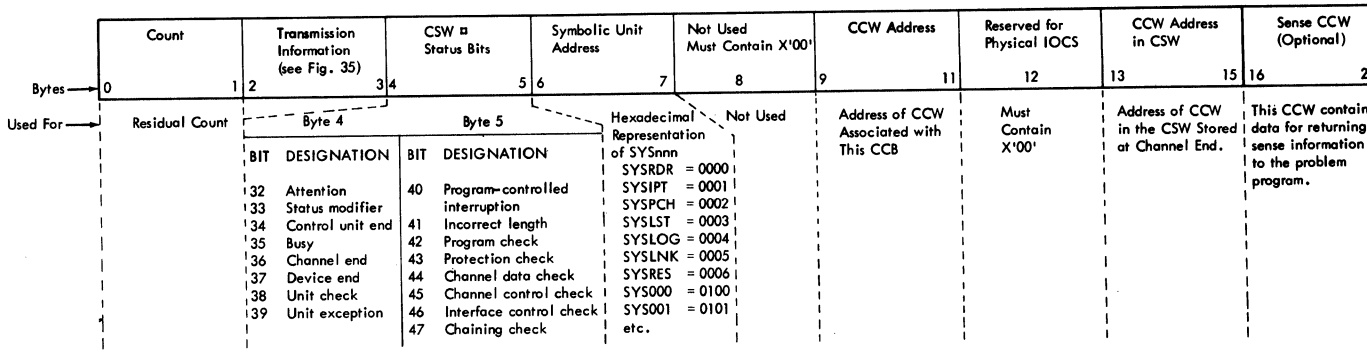
Name	Operation	Operand
[name]	EXCP	{Blockname / (1)}

The EXCP (execute channel program) macro instruction requests physical IOCS to star an input/output operation for a particular I/O device. The Blockname of the CCB established for the device is the only operand required in this instruction. Blockname can be specified as a symbol or in register notation.

Physical IOCS determines the device concerned, from the Command Control Block specified by Blockname, and places the Command Control Block (CCB) in a queue of such CCB's for this device. If the channe and device are available, the channel program is started. Program control is then returned to the problem program. I/O interruptions are used to process I/O completion and to start I/O for requests if the channel or device was busy at EXCP time.

WAIT MACRO

Name	Operation	Operand
[name]	WAIT	{Blockname / (1)}



¤ Bytes 4 and 5 contain the status bytes of the Channel Status Word (Bits 32 - 47). If byte 2, bit 5 is on and device end results as a separate interrupt, device end status will be ORed in.

●Figure 37. Command Control Block (CCB)

This macro instruction is issued whenever the program requires that an I/O operation, started by an EXCP instruction, be completed before execution of the problem program continues. For example, the transfer of data (a physical record) to main storage must be completed before that data can be added, moved to another area of main storage, or otherwise processed. When this WAIT instruction is executed in a batch job environment, processing is suspended until the traffic bit (byte 2, bit 0) of the related CCB is turned ON. Then programming automatically continues, and the data can be processed. In a multiprogramming environment, the Supervisor gives control to another program until the traffic bit is set ON.

The Blockname, as a symbol or in register notation, of the CCB established for the I/O device is the only operand required in this instruction. This is also the same name as that specified in the EXCP instruction for this device.

Before using the WAIT macro for a paper tape file, the user should refer to the section entitled Sequential Processing, Paper Tape Reader: Programming Considerations.

ALTERNATE TAPE SWITCHING

Alternate tape drives cannot be used on input processed by PIOCS. On output, automatic alternate switching can be accomplished by using the DTFPH and FEOV macro instructions. FEOV writes the standard trailer labels, and any user-trailer labels (if DTFPH LABADDR is specified). When the new volume is mounted and ready for writing, IOCS writes the standard header labels, and the user-standard header labels, if any.

BYPASSING IMBEDDED CHECKPOINT RECORDS ON TAPE

The checkpoint information saved is written as a set of magnetic tape records: a 20-byte header record, as many core-image records as required to save the necessary parts of core, and a 20-byte trailer record identical to the header. The format of the header and trailer record is:

Bytes	Contents
0-11	/// CHKPT //
12-13	The number, in binary, of core image records following the header.
14-15	The total number, in unpacked hexadecimal, of records following the header.
16-19	The serial number of the checkpoint.

If checkpoint sets are imbedded in a file being read with physical IOCS, they must be recognized and bypassed. On any mode input tape, checkpoint sets may be identified by the first 12 bytes of the header or trailer records. Note that when reading backwards, the checkpoint header will occupy the 20 low-order bytes of the input area.

When bypassing checkpoint sets, three methods are possible:

1. Go into a read loop (forward or backward) until the checkpoint trailer (header if backward) is encountered.
2. Extract the count from bytes 12-13 of the header (or trailer if backwards), add 2 to this, and forward space (or backspace) that number of records. Read commands could also be used.
3. Extract bytes 14-15 of the header (or trailer if backwards), pack and convert the field to binary, and forward space (or backspace) that number of records. Read commands could also be used.

When bypassing checkpoint sets on 7-track tapes in translate mode, only method 3 can be used and only forward space (or backspace) record commands (not reads) can be used. Reads would create data checks.

COMMAND CHAINING RETRY

If the user generates his system to support command chaining retry, he can utilize this option for his physical IOCS channel programs by setting CCB bit 7, byte 3 ON. If this bit is ON and an error involving retry occurs, the retry begins with the last CCW executed. If the bit is OFF, the entire channel program is reexecuted.

If a command chain is broken by a condition (such as wrong-length record or unit exception) that does not result in device error recovery by physical IOCS, the user can determine the address of the last CCW executed and, if necessary, restart at that point.

To obtain the address of the last CCW executed, subtract 8 from the address stored in bytes 13-15 of the CCB.

When the command chaining retry bit is ON, the user must move the address of the first CCW in the channel program to bytes 9-11 of the CCB before each EXCP issued. This is done to ensure that the correct address is there because physical IOCS modifies this field when retrying after an I/O error and never restores it to the original value.

Command chaining should not be used to read multiple records from SYSIPT or SYSRDR. The bit should never be ON for DASD channel programs.

DATA CHAINING

When using Data Chaining, each CCW should contain the command code of the operation being executed. (If the CCW's were formed by the Assembler, they contain this code automatically.) This is necessary to ensure proper I/O error recovery. Because recovery frequently depends on the command being executed, the command in the last CCW executed is often examined.

CHANNEL PROGRAM, DASD FILE PROTECTED DEVICE

If the DASD containing the file to be processed is file protected, the user must begin his DASD channel program with a Seek (hex command code 07). Any other seek in the program must be a Seek Head (hex command code 1B). As shown in the following chart, when executing the channel program, the Supervisor sets up three commands in the channel program that it builds: a seek that is identical to the user's seek, a set file mask that prevents any other 07 seeks from being executed, and a transfer in channel (TIC) that transfers control to the command following the user's seek.

DTFPH MACRO

When physical IOCS macro instructions (EXCP, WAIT, etc) are used in a program, DASD or tape files with standard labels need to be defined by DTFPH entries (DTF for a file handled by Physical IOCS). DTFPH must also be used for a checkpoint file on a 2311.

Checkpoint File on 2311: The following parameters can be used:

DEVADDR=SYSnnn	optional
DEVICE=2311	required
LABADDR=name	optional
MOUNTED=SINGLE	required
TYPEFLE=OUTPUT	required
XTNTXIT=name	does not apply

If a DASD or tape file with standard volume and file labels is processed, a DTFPH header card and detail cards may be used (Figure 38). This set indicates to IOCS that labels are to be read and checked (on input) or written (on output). The header card is punched with DTFPH in the operation field and the symbolic name of the file in the name field. The symbolic name may be seven characters long.

CCWADDR=Name

This parameter allows the user to utilize the CCB generated within the first 16 bytes of the DTFPH table. CCWADDR specifies the symbolic name of the first CCW to be used with the CCB generated within the DTFPH macro. This name must be the same as the name specified in the assembler CCW statement that constructs the channel command word.

If this parameter is omitted, the location counter value of the CCB-CCW table address constant will be substituted for the CCW address.

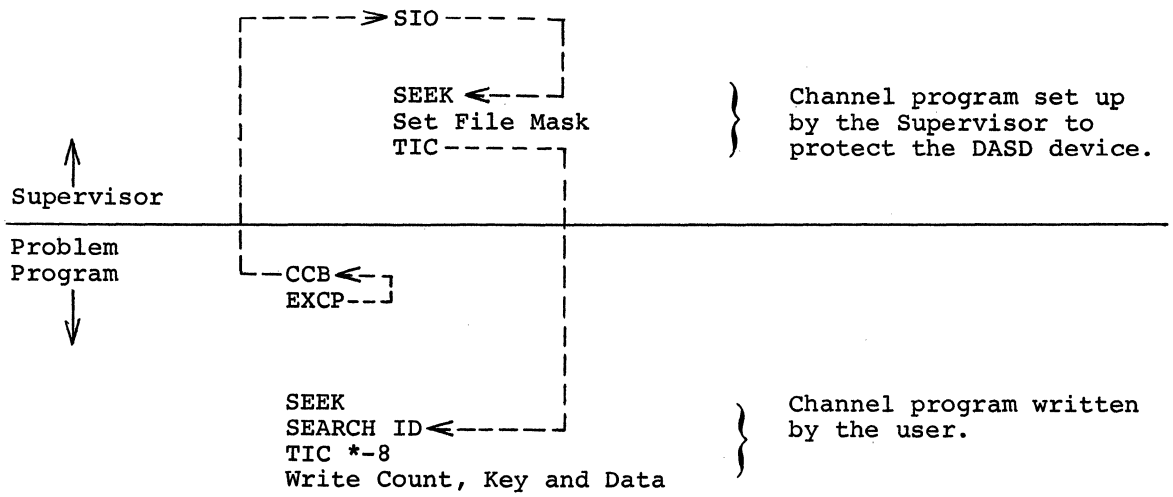
DEVICE={TAPE / 2311 / 2321}

If the file is contained on DASD, enter the proper identification: 2311 or 2321. If DEVICE=2311 or 2321, the DEVADDR entry can be omitted.

DEVADDR=SYSnnn

This entry specifies the symbolic unit (SYSnnn) to be associated with the logical file. The symbolic unit represents an actual I/O device address. The symbolic unit may be:

- SYSLNK for linkage-editing
- SYSPCH for main system punch device



- SYSLST for main system printer
- SYSIPT for main system input device
- SYSRDR for system reader
- SYSRES for system residence device
- SYS000-SYS244 for other units in the system

The symbolic unit (SYSnnn) is used in the Job Control ASSGN statement to assign the actual I/O device address to this file. The ASSGN card makes the file independent of the device it runs on. For example, a reel of tape may be mounted on any tape drive that is available at the time the job is ready to be run, merely by assigning that drive to the symbolic unit.

If SYSLST or SYSPCH are used as output tape units and alternate tape switching is desired upon detecting a reflective spot, the SEOV macro instruction must be used. (See SEOV.)

LABADDR=Name

The user may require one or more DASD or tape labels in addition to the standard file labels. If so, he must include his own routine to check (on input) or build (on output) the user label(s). He specifies the symbolic name of his routine in this entry, and IOCS branches to his routine after the standard label has been processed.

LABADDR may be included to specify a user routine for user header or trailer labels as follows:

- DASD input or output file: header labels only
- Tape input file: header labels only
- Tape output file: header and trailer labels

Thus, if LABADDR is specified for the file, user header labels can be processed for an input/output disk or tape file, and user trailer labels can be built for a tape output file. Similar to the functions performed by logical IOCS, physical IOCS reads input labels and makes them available to the user for checking, and writes output labels after they are built.

For a complete discussion of the LABADDR routine, see the OPEN sections of this manual.

MOUNTED=

This entry must be included for a DASD file to specify how many extents (areas) for the file are to be made available for processing when the file is initially opened. The entry must not be included for a tape file. One of the following specifications is entered after the = sign:

ALL — if all extents are to be available for processing. When the file is opened, IOCS checks all labels on all packs and makes available all extents specified by the user's control cards. Only one OPEN is

NAME	OPERATION	OPERAND	APPLIES TO							MUST BE INCLUDED	REMARKS
			2311 DISK DRIVE	2400 MAGNETIC TAPE UNIT	1442/2501/2520/ 2540 READER	1442/2520/2540 PUNCH	1403/1404/1443/ 1445 PRINTER	1052 PRINTER-KEYBOARD	2321 Data Cell Drive		
Filename	DTFPH		X	X					X	If DASD/Tape File with Standard Labels, DASD Files on File Protected DASD devices, or 2311 Checkpoint File	Specify 7- character or less Symbolic Filename.
		CCWADDR = Name	X	X					X	If the CCB generated by the DTFPH is to be used	Symbolic Name of User's First CCW
		DEVICE = {Tape / 2311 / 2321}	X	X					X	For each File	
		DEVADDR=SYSnnn		X						For tape File with Standard els Labels	Symbolic Unit for the Device Used for the File.
		LABADDR=Name	X	X					X	To Check/Build User-Standard Labels	Symbolic Name of User's Label Routine. For Tape Input and DASD Files, Applies to Header Labels Only.
		MOUNTED=ALL / SINGLE	X						X	For Each DASD File	All Extents Are to Be Available at the Initial OPEN. Only the First Extent Is to Be Available at the Initial OPEN.
		TYPEFLE=INPUT / OUTPUT	X	X					X	For DASD/Tape Input File with Standard Labels / For DASD/Tape Output File with Standard Labels 2311 Checkpoint File	
		XTNTXIT=Name	X						X		Symbolic Name of User's Extent Routine.

Figure 38. DTFPH Entries

required for the file. ALL should be specified whenever the user plans to process records in a manner similar to that performed by the direct access method. In any case, the user must supply a LBLTYP card.

SINGLE if only the first extent on the first volume is to be available for processing. SINGLE should be specified when the user plans to process records in sequential order. IOCS checks the labels on the first pack, and makes the first extent specified by the user's control cards available for processing. The user must keep track of the extents and issue a subsequent OPEN whenever another extent is required for processing. The user will find the information in the DTFPH tabl helpful in keeping track of the extents:

DTFPH table (referenced by Filename)

Byte	Contents
0-15	CCB (Symbolic unit has bee initialized in the CCB)
54-57	Extent Upper Limit (CCHH)
58-59	BB Seek Address
60-63	Extent Lower Limit (CCHH).

On each OPEN after the first, IOC makes available the next extent specified by the control cards.

When the user issues a CLOSE for an output file, the volume on which he is currently writing records will be indicated, in the file label, as the last volume for this file.

TYPEFLE={Input / Output}

This entry must be included to specify the type of file (input or output). One specification or the other is entered immediately after the = sign.

XTNTXIT=Name

This entry is included if the programmer wants to process XTENT card information. It specifies the symbolic name of the user's extent routine. The DTFPH entry MOUNTED=ALL must also be specified for the file.

Whenever XTNTXIT is included, IOCS branches to the user's routine during the initial OPEN for the file. It branches after each specified extent has been completely checked and after conflicts, if any, have been resolved.

Upon entry to the user's routine, IOCS stores in register 1 the address of a 14-byte area from which the user can retrieve extent card information (in binary form). This area contains:

Bytes	Contents
0	Extent type code (as specified in the XTENT card)
1	Extent sequence number
2-5	Lower limit of the extent (CCHH)
6-9	Upper limit of the extent (CCHH)
10-11	Symbolic unit
12	Old bin number
13	Present bin number of the extent (B2)

The user returns to IOCS by use of the LBRET macro instruction.

The Supervisor is a control program that provides specialized services to problem programs. These services differ slightly, depending on the execution environment. In the batch-job environment, the Supervisor processes interruptions, I/O requests, and program retrieval. In addition to these functions, in the multiprogramming environment the Supervisor also determines which program (foreground1, foreground2, or background) is to be executed.

The interruptions handled by the Supervisor result from five conditions:

- Input/Output
- Program Check
- Machine Check
- External Signal (including timer)
- Supervisor Call

The user can request the Supervisor to set up linkages so that his routines can handle program check, and operator-communication and/or timer interrupts.

The Supervisor also contains a communication region (see Figure 39) that problem programs can use for storing information between job steps. There are no restrictions on the use of this area in the batched-job environment. In the multiprogramming environment any program can read from this region, but only background programs can write in it.

Several macro instructions are available to the programmer to enable him to communicate with the Supervisor. Thus, he can utilize the functions performed by the Supervisor or have access to the communication region in the Supervisor. To make use of the Supervisor functions requires switching from problem state to Supervisor state. Therefore, most macro instructions used for this purpose generate a Supervisor Call (SVC) instruction. The macro instructions included in this section are:

LOAD	Loads a program phase and returns control to the calling phase.
FETCH	Loads and gives control to a program phase.
COMRG	Obtains the address of the communication region. This macro does not generate an SV
MVCOM	Modifies the content of the user's portion of the communi cation region.
GETIME	Obtains the time of day. Thi macro does not generate an SV
SETIME	Requests the Control Program to take a program exit or set a bit in the TECB after a spe cific time interval.
STXIT	Activates a user's program check, interval timer, or operator communication routine, or cancels the use of such a routine.
EXIT	Returns to the point of inter ruption from a user routine for interval timer, program check, or operator communication.
TECB	Generates a Timer Event Contr Block.
WAIT	Yields control until the expiration of the interval time
PDUMP	Obtains a selective (snapshot dump of main storage.
DUMP	Terminates the job step and provides a dump of main storage.
CANCEL	Terminates all remaining step of the job.
EOJ	Informs the Supervisor that t current problem-program job step has been completed.
CHKPT	Causes checkpoints to be take in a batch or background program.

Multiprogramming Restrictions on Use of Supervisor Macros: If MVCOM is used in a foreground program, the program is cancele because the bytes in the communication region that can be modified by this macro do not contain information for foreground

programs. The interval timer macros SETIME, STXIT IT, and EXIT IT can be used in only one program (foreground1, foreground2, or background) at a time. This program is specified at system-generation time but can be changed by the operator. CHKPT is ignored in a foreground job.

PROGRAM LOADING

Phases may be loaded into main storage from the Core Image Library with the FETCH and LOAD macro instructions. FETCH gives control to the phase that was loaded while LOAD returns control to the phase that issued the macro instruction. Self-relocating phases must be loaded using the LOAD macro instruction (rather than the FETCH) with the load address specified in a register.

FETCH--FETCH A PHASE

Name	Op	Operand
[name]	FETCH	{phasename / (1)} [,{entryname / (0)}]

The FETCH macro instruction loads the phase specified in the first parameter. The phase name can be 1-8 characters long. Control is passed to the address specified by the second operand. If the second operand is not specified, control is passed to the entry point determined at linkage-edit time.

The parameters can be specified either as symbols or in register notation. When register notation is used for phasename, the register must be preloaded with the address of an eight-byte field that contains the phasename as alphameric characters. If necessary, the phasename should be padded with blanks.

If ordinary register notation is used for entryname, the absolute address of the entry point of the phase should not be preloaded into register 1. If, instead, a symbolic name is used for entryname, the macro expansion results in a V-type address constant. The entryname does not have to be identified by an EXTRN statement.

LOAD--LOAD A PHASE

Name	Op	Operand
[name]	LOAD	{phasename / (1)} [,{loadaddr / (0)}]

The LOAD macro instruction loads the phase specified in the first parameter and returns control to the calling phase. The phasename can be 1-8 characters long. The user should code his LOAD in a place where it cannot be overlaid by the new phase.

After execution of the macro, the entry-point address of the called phase is returned to the programmer in register 1. This entry-point address is determined at linkage-edit time.

If an optional address parameter is provided, the load-point address specified to the linkage editor is overridden, and the phase is loaded at the address specified. The address used must be outside the Supervisor area. When an overriding address is given, the entry-point address is relocated and returned in register 1. None of the other addresses in the phase are relocated.

The parameters can be specified either as symbols or in register notation. When register notation is used for phasename, the register must be preloaded with the address of an eight-byte field that contains the phasename. If necessary, the phasename should be left-justified and padded with blanks. If ordinary register notation is used for loadaddr, the parameter should not be preloaded into register 1.

COMMUNICATION REGION

As shown in Figure 39 and described here, the communication region is a 46-byte storage area within the Supervisor. Batch-jobs can read and write in this region. In a multiprogramming environment, all programs can read the data located in this area, but only background programs can write there.

Field Length	Information
8 bytes	Calendar date. Supplied from system date whenever a JOB statement is encountered. The field can be two forms: MM/DD/YY or DD/MM/YY where MM is month, DD is day, and YY is year. It can be temporarily overridden by a DATE statement.
2 bytes	Address of first byte of problem program area for the batch or background job.
2 bytes	Reserved for control program use.

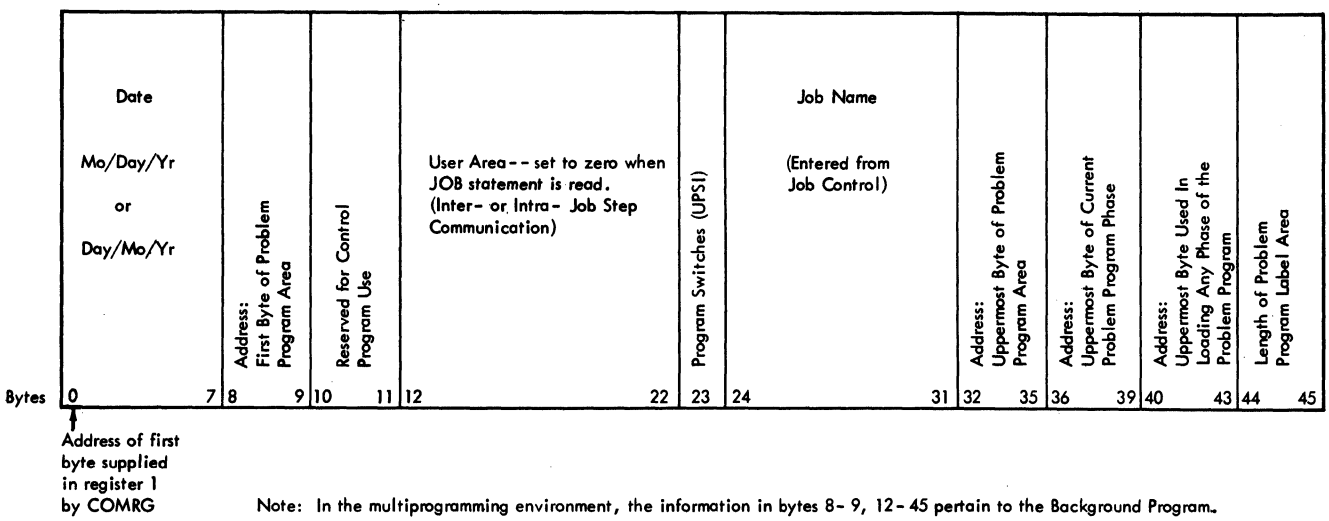


Figure 39. Communication Region (in Supervisor)

11 bytes — User area for inter- or intra-job step communications. All 11 bytes set to zero when JOB statement for the batch or background job is encountered.

1 byte — UPSI (user program switch indicators). Set to binary zero when JOB statement for the batch or background job is encountered. Initialized by UPSI Job Control statement.

8 bytes — Job name as found in the JOB statement for the batch or background job.

4 bytes — Address of the uppermost byte of the batch or background program area. When the first phase of a foreground or background program is loaded and given control, register 2 contains the address of the uppermost byte of the respective program area.

4 bytes — Address of the uppermost byte of the current phase placed in the problem-program area by the last FETCH or LOAD macro instruction in the batch or background job.

4 bytes — Address of the uppermost byte used in loading any phase of the batch or background job. This value may be incorrect if the program (using the LOAD macro) loads a phase above its linkage-edited origin, or all phases of the program are not linkage-edited together.

2 bytes — Length of batch or background program label area.

Macro instructions (COMRG and MVCOM) are provided to allow the problem program to communicate with the Supervisor and the communication region. A brief discussion of these macro instructions follows.

COMRG--GET ADDRESS OF COMMUNICATION REGION

Name	Operation	Operand
[name]	COMRG	

When COMRG is issued, the address of the communication region is placed in register 1. Any problem program can read any portion of the communication region by using register 1 as a base register.

MVCOM--MOVE TO COMMUNICATION REGION

Name	Operation	Operand
[name]	MVCOM	to,length,{from / (0)}

The MVCOM macro instruction is used to modify the content of bytes 12-23 of the

communication region. This macro cannot be used in a foreground program.

The operand from represents the address, either as a symbol or in register notation, of the bytes to be inserted; length represents the number of bytes (1-12) to be inserted; to is the relative address of the first communication region byte to be modified (12-23). (The to address used is relative to the first byte of the region.)

The following example shows how to move three bytes from the symbolic location DATA into bytes 16-18 of the communication region.

Name	Operation	Operand
[name]	MVCOM	16,3,DATA

TIME OF DAY MACRO

GETIME--GET TIME OF DAY IN REGISTER 1

Name	Operation	Operand
[name]	GETIME	{STANDARD / BINARY / TU}

The GETIME macro instruction is used to obtain the time of day at any time during program execution. STANDARD is assumed if no operand is given.

If STANDARD is specified, the time of day is returned in register 1 as a packed decimal number: HHMMSS (where H is hours, M is minutes, and S is seconds) with low-order sign. The time of day may be stored, unpacked, or edited.

Note: Lengthy conversion routines are generated (in line) each time STANDARD is used. Therefore, this function should be put into a subroutine if it is used frequently.

If BINARY is specified, the time of day is returned in register 1 as a binary integer in seconds.

If TU is specified, the time of day is returned in register 1 as a binary integer in units of 1/300 second.

GETIME can be used only if the timer feature was specified at system generation time and if the CPU has the timer feature.

Note: The timer feature is independent of the interval timer options (SETIME and STXIT). GETIME can be used by any area in a multiprogramming environment, regardless of which area is using the timer.

INTERVAL TIMER AND USER EXIT MACROS

Programs using the interval timer macros--SETIME, WAIT, TECB, STXIT IT, EXIT IT--must be executed with a Supervisor containing the optional interval timer routines and must be executed on a CPU having the timer feature. The user specifies at system-generation time whether the Supervisor is to be generated with the interval timer routines.

In a multiprogramming environment, only one program at a time can use the interval timer macros. This program is specified at system-generation time but can be temporarily changed by the operator.

There are two distinct methods of using the interval timer macros. Only one method can be used at a time. The first method allows the user to set the timer and enter a routine in his program when the time elapses. The SETIME, STXIT, and EXIT macros are used to do this. In the second method, a given routine can be performed at timed intervals. The SETIME, TECB, and WAIT macros are used. The time set is a real-time interval and is not stopped or adjusted when the program using the timer does not have control. This should be noted by the lower-priority programs in a multiprogramming environment.

METHOD-1 MACROS

SETIME--SET INTERVAL TIMER

Name	Operation	Operand
[name]	SETIME	{seconds / (1)}

The SETIME macro instruction is used to set the interval timer to the value that is specified in the operand. The largest allowable value is 55924, which is equivalent to 15 hours, 32 minutes, 4 seconds. A register may be specified as the operand. The register must contain the number of seconds in binary. When the specified timer interval has elapsed, the interval-timer routine supplied by the user will be entered.

If a routine has not been supplied to the Supervisor (via the STXIT macro instruction) by the time of the interruption, the interruption will be ignored.

When a program is restarted from a checkpoint, any timer interval set by a SETIME macro is not restarted.

STXIT--SET LINKAGE TO USER ROUTINE(S)

Name	Operation	Operand
to establish linkage		
[name]	STXIT	{PC / IT / OC}, {rtnaddr / (0)}, {savarea / (1)}
to terminate linkage		
[name]	STXIT	{PC / IT / OC}

The STXIT (set exit) macro instruction is used to establish or terminate a linkage from the Supervisor to a user's routine for Interval Timer, Operator, or Program-Check interrupt handling. If only the first operand is present, linkage to the user's routine is terminated.

The operands are described as follows.

PC — Program Check Interruption

IT — Timer Interruption

OC — Operator Attention Interrupt

rtnaddr — Entry-point Address of the routine that handles the interruption described in the first operand.

savarea — Address of a 72-byte area in which the Supervisor stores the old PSW and general registers 0-15. In the case of stacked interrupts, the user must have a separate save area for each routine.

The routine address and the savarea address can be given in register notation. However, the routine address should not be specified in register 1.

In a batch or background job, the OC routine is entered when the external interrupt key on the console is pressed. In a foreground program, the OC routine is entered when the request key on the 1052 is pressed and the foreground OC routine is requested. The IT routine is entered when the interval timer elapses. The PC routine is entered when a program check occurs. If a PC occurs in a routine that is being executed from the Logical Transient Area, the job containing the routine is canceled.

In all cases if a STXIT macro is given and the Supervisor was not generated to handle the requested facility, the job is canceled. This also applies to a program that requests the timer interrupt and is not allocated the timer.

If a timer or operator-attention interrupt occurs and linkage has not been estab lished to a user routine, the interrupt is ignored. If a Program Check occurs withou exit linkage established, the program is terminated.

The following chart shows what happens when an interrupt occurs while an interrup routine is being processed.

Routine Being Processed	Interrupt Occuring		
	PC	OC	IT
PC	C	H	H
OC	H	I_b E_f	H
IT	H	H	I

C — Job canceled.

H — Interrupt honored. When processing of new routine completes, control returns to interrupted routine.

E_f — Error message given in foreground program and control returns to interrupte OC routine.

I — Interrupt ignored for all programs

I_b — Interrupt ignored in batch job or back ground program.

Notes:

1. When restarting a program from a checkpointed position, any STXIT linkages established prior to the checkpoint are destroyed.

2. If a program is using a logical transient routine when a timer interrupt occurs, the user timer routine is not entered until the logical transien routine is released.

EXIT--EXIT FROM USER'S INTERRUPT ROUTINE(S

Name	Operation	Operand
[name]	EXIT	{PC / IT / OC}

The EXIT macro instruction is used to retu from a user routine, specified in the STXI macro instruction, to the point in the interrupted program where the interruption occurred. The PSW and registers are restored from the savarea; hence the savarea

contents should not be destroyed. The operands have the following meanings:

PC Exit from the user's program-check routine.

IT Exit from the user's interval-timer routine.

PC Exit from user's routine that handles the operator-attention interrupt.

METHOD-2 MACROS

TECB-BUILD TIMER EVENT CONTROL BLOCK

Name	Operation	Operand
tecbname	TECB	

The TECB causes a Timer Event Control Block, shown in Figure 40, to be generated at the address of tecbname. This block contains an event bit that is set to indicate when the time interval specified in SETIME has elapsed.

SETIME-SET INTERVAL TIMER

Name	Operation	Operand
[name]	SETIME	{seconds / (1)}, {tecbname / (r)}

The SETIME macro sets the amount of time that must elapse before the TECB event bit is set to 1 and the routine following the WAIT macro can be processed. When SETIME is issued, the event bit is set to 0.

The number of seconds can be specified directly or in register notation. The largest allowable value is 55924, which is equivalent to 15 hours, 32 minutes, 4 seconds. If a register is specified, the register must contain the number in binary.

The user can specify the tecbname or specify the register (r) (r cannot be 0 or 1) in which he has placed the address of the corresponding TECB. After SETIME is executed, the Supervisor returns the TECB address in register 1.

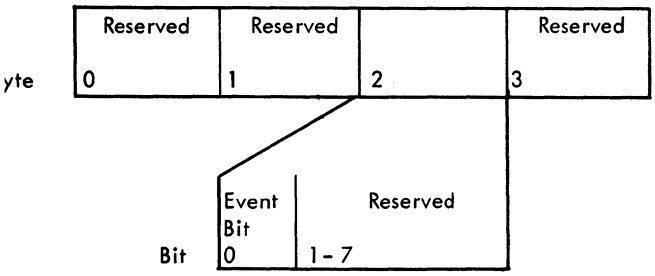
WAIT--WAIT FOR TIMER ELAPSE

Name	Operation	Operand
[name]	WAIT	{tecbname / (1)}

The WAIT macro is used to ensure that the time interval specified in SETIME has elapsed (event bit turned ON) before execution of the program issuing the WAIT continues. When a WAIT macro is processed in a multiprogramming environment, control is given to the Supervisor, which makes the time available to a lower-priority program.

The user can either specify the tecbname or use register notation. The WAIT macro instruction loads the TECB address into register 1 unless register (1) is specified.

Note: The SETIME macro instruction leaves the TECB address in register 1.



he Event Bit is set ON by the Supervisor's Timer Routines

alue	Indication
0	time specified in SETIME has not elapsed
1	time specified in SETIME has elapsed

igure 40. The Timer Event Control Block (TECB)

THE DUMP MACROS

PDUMP--PARTIAL DUMP OF MAIN STORAGE

Name	Operation	Operand
[name]	PDUMP	{address1 / (r)}, {address2 / (r)}

This macro instruction provides a hexadecimal dump of the general registers and of the main storage area that is contained between the two address expressions (address1 and address2). One or both of the addresses can be given in registers. Special register notation is not necessary for use in a self-relocating program. The contents of registers 0-1 are destroyed, but the CPU status

is retained. Thus, PDUMP furnishes a dynamic dump (snapshot) useful for program checkout. Processing continues with the next user instruction.

In a batch or background job, this dump is directed to SYSLST. When SYSLST is a 2311, the user must issue an OPEN macro after each PDUMP that is executed. This OPEN macro updates the disk address maintained in the DTF table to agree with the address where the PDUMP output ends. If the OPEN is not issued, the address is not updated, and the program is canceled when the next PUT is issued.

In a foreground job the dump is directed to SYS000, which can be either a printer or a tape. The records are 121 bytes long; the first byte is an ASA control character.

DUMP--DUMP MAIN STORAGE

Name	Operation	Operand
[name]	DUMP	

This macro terminates the job step and gives a hexadecimal dump.

In a batch-job or a background program, the Supervisor, the batch or background program, and the general registers are dumped onto SYSLST.

In a foreground program, the dump goes to SYS000 and contains the supervisor, the foreground program that issued the macro, and the general registers. SYS000 can be either a printer or a tape. (Before the macro is issued, the tape must be opened, if necessary, and positioned as desired. The records are 121 bytes long; the first byte is an ASA control character.)

THE CANCEL AND EOJ MACROS

CANCEL--CANCEL THE JOB

Name	Operation	Operand
[name]	CANCEL	

This macro instruction causes the job to be terminated. No dump of main storage is provided.

EOJ--END-OF-JOB STEP

Name	Operation	Operand
[name]	EOJ	

The EOJ macro instruction is issued at the end of a problem-program step to inform the system that the job step is finished. The operand field is ignored.

CHECKPOINTING A PROBLEM PROGRAM

Checkpoint is a means of recording the status of a problem program at desired intervals. Restart is a means of resuming the execution of the program from one of the checked points rather than from the beginning, if processing is terminated for any reason before the normal end of program For example, a job of higher priority may require immediate processing, or some malfunction such as a power failure may occur and cause an interruption. The checkpoint ability is provided through the CHKPT macro while the restart ability is provided through Job Control. (For information on restarting a checkpointed program, see the System Control and Service publication listed on the front cover.)

USE of CHKPT MACRO

The CHKPT macro can be issued by any program in a batch-job environment or by a background program in a multiprogramming environment. It will be ignored in a foreground program. CHKPT will also be ignored under the following conditions:

1. The device on which the checkpoint records are to be written is not a magneti tape or a 2311 disk. (The device must be a 2311 disk if filename operand is present.)
2. End-of-reel is detected while writing the checkpoint on tape.
3. The area on disk is not large enough for a single checkpoint.
4. The macro is issued by a telecommunication program that has any I/O operation(s) pending on a telecommunication device.
5. The user-specified end address is greater than the end of the background-job area.

6. The CHKPT macro is issued before the disk checkpoint file is opened.

7. Any of the required DTFPH parameters for the disk checkpoint file contain errors.

CHKPT MACRO

Name	Operation	Operand
[name]	CHKPT	SYSnnn, {restart address / (r1)} [,end address (r2)] [,tpointer (r3)] [,dpointer (r4)] [,Filename (r5)]

SYSnnn specifies the logical unit on which the checkpoint information will be stored. It must be a magnetic tape or a 2311 disk. (See Checkpoint File.)

Restart address (or r1) specifies a symbolic name of the problem program statement (or register containing the address) at which execution is to restart if processing must be continued later.

End address (or r2) is a symbolic name (or register containing the address) of the uppermost byte of the problem program area required for restart. This address must follow the logic modules being included from the relocatable library.

If this operand is omitted, all of main storage above the Supervisor will be checkpointed in a batch-job environment, and all of the background program area in a multiprogramming system.

This operand provides two advantages. One, less time and space is required for recording the checkpoint record set. Two, if a program using 32K of storage is being run in a larger system and only 32K is checkpointed, that program can be restarted either on a 32K system or as a 32K background job in a multiprogramming system.

Tpointer (or r3) is the symbolic name of an eight-byte field contained in the problem program area. (See Repositioning Magnetic Tape.)

Dpointer (or r4) is the symbolic name of a DASD operator verification table that the user can set up in his own area of main storage. (See DASD Operator Verification Table.)

Filename (or r5) is used only for checkpoint records on disk. It is the name of the associated DTFPH macro. (See Checkpoint on Disk.)

Special Register Notation cannot be used with any of these operands.

Information that Is and Is Not Saved

When the CHKPT macro is issued, the following information is saved:

- Information for the Restart and other Supervisor or Job Control routines.
- The general registers.
- Bytes 8-10 and 12-45 of the communication region.
- The problem program area (see End-Address Operand).
- All DASD file protection extents attached to logical units belonging to the checkpointed program.

Note: If the program is using DASD system input or output files (SYSIPT, SYSLST), they must be reopened in the user restart routine to obtain the current DASD address.

The following information is not saved:

- The floating point registers. (If needed, these registers should be stored in the problem program area before issuing CHKPT, and restored in a user restart routine.)
- Any linkages to user routines set by the STXIT macro. (If needed, STXIT should be used in user's restart routine.)
- Any timer values set by the SETIME macro. (If needed, SETIME should be used in user's restart routine.)
- The program mask in problem program PSW. (If other than all zeros is desired, the mask should be reset in user's restart routine.)

Note: A user's restart routine must also reopen any DASD system input or output files (SYSIPT, SYSLST, etc) that are used.

CHECKPOINT FILE

The checkpoint information must be written on a 2311 disk or a magnetic tape--either 7 or 9 track. The 7-track tape can be in either data conversion or translation mode; however, the magnetic tape unit must have the data conversion feature. On 7-track tapes, the header and trailer labels are written in the mode of the tape; the data records are written in data convert mode, odd parity.

Checkpoints on Tape

The programmer can either establish a separate file for checkpoints or imbed the checkpoint records in an output data file. When the data file is read at a later time using logical IOCS, the checkpoint records are automatically bypassed. If physical IOCS is used, the user must program to bypass the checkpoint record sets. See Processing Records with Physical IOCS.

If a separate magnetic tape checkpoint file with standard labels is maintained, the labels should be either checked by an OPEN or bypassed by an MTC command before the first checkpoint is taken.

Checkpoints on Disk

If checkpoints are written on a 2311 disk, the following must be observed:

- One continuous area on a single pack must be defined at execution time by the Job Control cards necessary to define a DASD file.
- The number of tracks required is computed as follows:

$$N\left[\frac{X \quad Y}{1+\overline{30}+\overline{20}} + \frac{C}{3625}\right]$$

where N = the number of sets of checkpoint records to be retained. (When the defined extent is full, the first set of checkpoint records is overlaid.)

C = the number of bytes to be checkpointed in the user's problem program up to the end address specified in the CHKPT macro operand.

X = the number of 2311 XTENTS including nonoverlapping split-cylinder XTENTS. If split-cylinder XTENTS overlap on the same cylinder the number of XTENTS counted is one use by the program. (This number is zero if DASD file protect is not used.

Y = same as above for 2321.

For each division, the remainder i rounded up to the next highest whc number before multiplying by N.

- Each program can use a common checkpoint file or define a separate one. If a common file is used, only the las program using the file can be restarte
- The checkpoint file must be opened before the CHKPT macro can be used.
- A DTFPH macro must be included for use by OPEN and the checkpoint routine. See Processing Records with Physical IOCS: DTFPH Macro.

REPOSITIONING I/O FILES

The I/O files used by the checkpointed prc gram must be repositioned on restart to th record that the user wants to read or writ next. Checkpoint provides no aids for repositioning unit-record files. The programmer must establish his own repositioning aids and communicate these to the operator when necessary. Some suggested ways are:

- Taking checkpoints at a logical break point in the data, such as paper tape end of reel.
- Switching card stackers after each checkpoint.
- Printing information at checkpoint to identify the record in process.
- Issuing checkpoints on operator demand

User sequential DASD input or output files require no repositioning.

When updating DASD records in an existing file, the programmer must be able to identify the last record updated at checkpoint in case he needs to restart. This can be done in various ways, such as:

- Creating a history file to record all updates.
- Creating a field in updated records to identify the last transaction record that updated it. This field can be compared against each transaction at restart time.

epositioning Magnetic Tape

heckpoint provides some aid in reposition-
ng magnetic tape files at restart. Files
an be repositioned to the record following
he last record processed at checkpoint.

The following discussion narrates an
xample, in chart form, given at the end of
his section. The fourth operand of the
HKPT macro points to two V-type address
onstants that the user specifies in his
oding. The order of these constants is
mportant.

- The first constant points to a table containing the filenames of all the logical IOCS magnetic tape files that are to be repositioned.
- The second constant points to a table containing repositioning information for physical IOCS magnetic tape files that are to be repositioned.
- If the first, second, or both constants are zero, no tapes processed by logical, physical, or both types of IOCS, respectively, will be repositioned.

f the tables are contained in the same
SECT as the CHKPT macro, the constants may
e defined as A-type constants.

The user must build the tables discussed.
ach filename in the logical IOCS table
oints to the corresponding DTF table where
OCS maintains repositioning information.
he user should note the following:

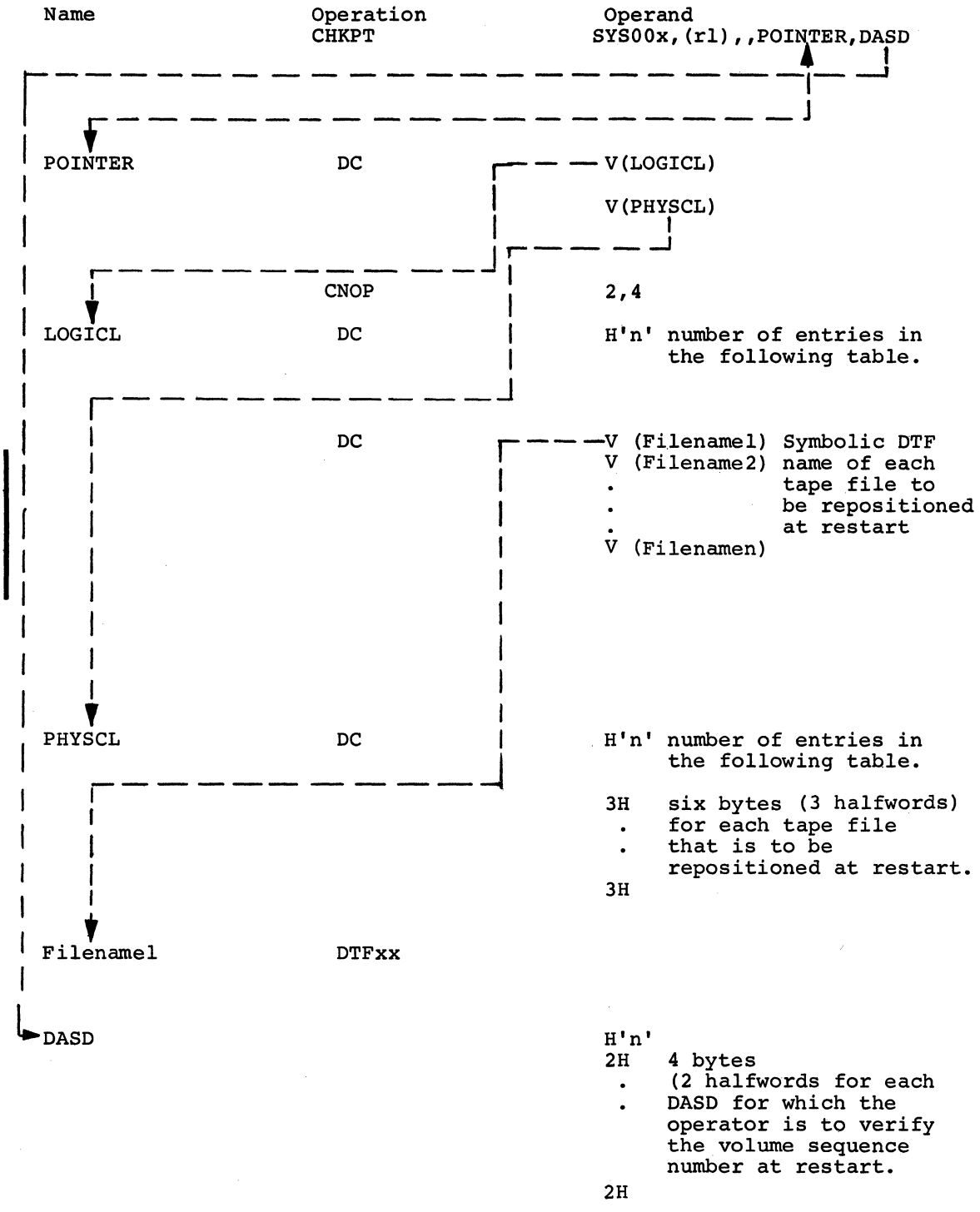
- Magnetic tapes with nonstandard labels should be repositioned past the labels at restart time (presumably the labels are followed by a tapemark so that forward-space file may be used).
- If a tape, that is to be repositioned, is processed with nonstandard labels and read backwards, the user must keep a physical IOCS repositioning table, because for this case, the physical record count kept by IOCS will be incorrect. The physical record count must be the number of forward reads necessary for restart to position the tape.
- Restart does not rewind magnetic tapes when repositioning them.
- A multifile reel should be prepositioned to the beginning of the desired file.
- The correct volume of a multivolume file must be mounted for restart.
- For tapes with a standard VOL label, restart writes the file serial number and volume sequence number on SYSLOG, and gives the operator the opportunity to verify that the correct reel is mounted.
- IOCS can completely reposition files on system logical units (SYSIPT, SYSLST, etc), if the tape is not shared with any other program and if the user keeps a physical IOCS repositioning table. However, if a system logical unit file is shared with other programs, a problem exists. Output produced after the checkpoint is duplicated at restart. Input records must be reconstructed from the checkpoint, or the user restart routine must find the last record processed before checkpoint.

The entries in the physical IOCS table are as follows:

First halfword--hexadecimal representation of the symbolic unit address of the tape (copy from CCB).

Second halfword--number of files within the tape in binary notation. That is, the number of tapemarks between the beginning of tape and the position at checkpoint.

Third halfword--number (in binary notation) of physical records between the preceding tapemark and the position at checkpoint.


Name
Operation
CHKPT
Operand
SYS00x,(r1),,POINTER,DASD
POINTER
DC
V(LOGICL)
V(PHYSCL)
CNOP
2,4
LOGICL
DC
H'n' number of entries in the following table.
DC
V (Filename1) Symbolic DTF
V (Filename2) name of each
. tape file to
. be repositioned
. at restart
V (Filenamen)
PHYSCL
DC
H'n' number of entries in the following table.
3H six bytes (3 halfwords)
. for each tape file
. that is to be
repositioned at restart.
3H
Filename1
DTFxx
DASD
H'n'
2H 4 bytes
. (2 halfwords for each
. DASD for which the
operator is to verify
the volume sequence
number at restart.
2H

DASD OPERATOR VERIFICATION TABLE

If the Dpointer operand is used, the user can build a table, in his own area of main storage, to provide the symbolic unit number and the bin number of each DASD file used by his program. At restart, the volume sequence number of these files will be printed on SYSLOG, and the operator can verify them.

The entries in the DASD Operator Verification Table must consist of the following two halfwords, in the order stated:

- The symbolic unit in hexadecimal notation.
- The bin number in hexadecimal notation.

There must be one entry for each DASD unit that is to be verified by the operator.

A program may consist of several phases which have been produced by the same language translator or different language translators (FORTRAN or COBOL, for example). These phases are then combined by the Linkage Editor. This process is meaningful only if one phase can branch to another phase and deliver parameters to it. The routine called should also be able to provide the calling routine program which results.

If control passes from one routine to another within the problem program, the linkage between the routines is referred to as direct linkage. In direct linkage, no linkage to the Supervisor is involved. Three macro instructions, using conventions of the Assembler language, are used for linkage between routines: CALL, SAVE, and RETURN.

Linkage between the main program and two subroutines is shown in Figure 41. Linkage can proceed through as many levels as required, and each routine may be called from any level. In the standard direct linkage, a routine always returns to the next higher level.

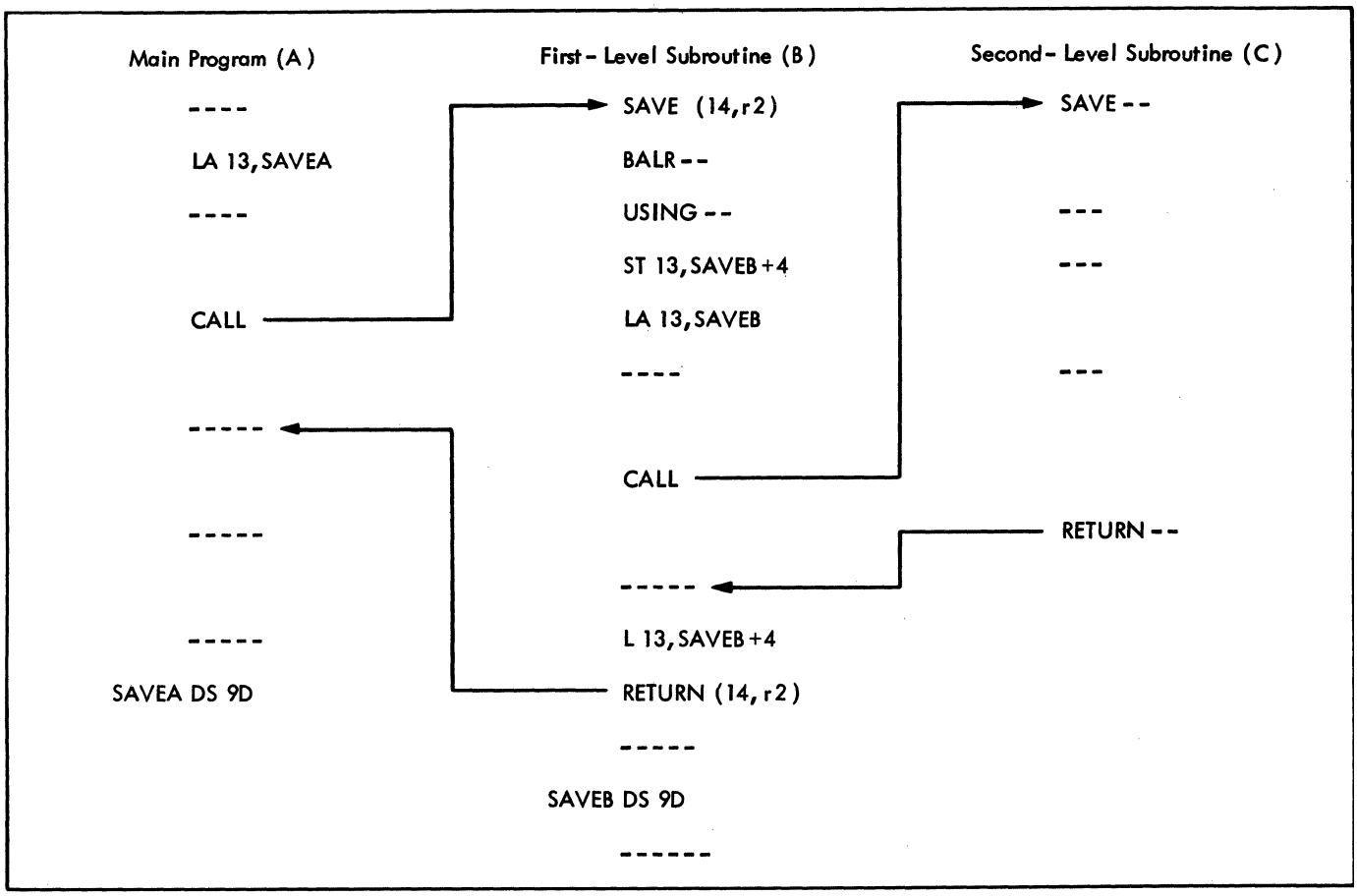


Figure 41. Direct Linkage

LINKAGE REGISTERS

The registers having specific roles in linkage are listed in Figure 42. The greatest use of linkage registers occurs with direct linkage. However, linkage registers are also used in linkages with the Supervisor, which are normally achieved through macro instructions.

Some of the linkage register identities and uses are shown in the following typical direct linkage calling sequence:

```
          CNOP  2,4
          LA    13,SAVEAREA     load save
                                area address
          LA    14,RETADR       load return
                                address
          L     15,=V(SUBR)     load entry
                                point addr
          BALR  1,15            load para-
                                meter list
                                address
          DC    A(PAR1,PAR2)    parameter
                                list

RETADR    ...
SAVEAREA  DS    9D
```

In the above sequence, a higher level program (the calling program) gives control to a lower level program (the called program) by branching to the address in register 15. Register 15 is the entry point register; it can be used to provide initial addressability in the called program.

Before branching, the calling program loads register 14, the return register, with the address to which the called program should return control. Two parameters, PAR1 and PAR2, are passed to the called program, in a list pointed to by register 1, which is the parameter list register.

SAVE AREA USE

Registers used by a called program must have their contents saved and restored by each lower level program that is given control by a higher level program. This conserves main storage, because the instructions to save and restore registers need not be in each calling sequence in the higher level program. The save area used is in the higher level program and has a standard format so that all programs can save registers in a uniform manner. Save areas are chained together in ascending order so that register contents can be restored as control is returned to the higher level programs. Save areas can also optionally be chained together in descending order.

REGISTER SAVING AND RESTORING RESPONSIBILITIES

Every program, before it executes a direct linkage, must provide a save area and place the address of this save area in register 13. A program can use the same save area

REGISTER NUMBER	REGISTER NAME	CONTENTS
0	Parameter register	Parameters to be passed to the called program.
1	Parameter register or Parameter list register	Parameters to be passed to the called program. Address of a parameter list to be passed to either the control program or a user's subprogram.
13	Save area register	Address of the register save area to be used by the called program.
14	Return register	Address of the location in the calling program to which control should be returned after execution of the called program.
15	Entry point register	Address of the entry point in the called program.

Figure 42. Linkage Registers

for all of its linkages. Unless the program requires register 13 for other reasons, the address of the save area can be loaded into it once, when the program is entered.

The save area in the calling program is used by a called program in direct linkage to save the contents of registers the called program will use. Register saving should be accomplished by using a SAVE macro instruction. Because register saving should be the first action taken by the called program, the SAVE macro instruction should be used at the entry point of the called program. The called program should use a RETURN macro instruction to return control to the calling program, and to restore the saved registers from the save area.

Words 6 through 18 of the save area in the calling program may be used by the called program to save registers, or as a work area.

SAVE AREA

A save area occupies 9 double-words and is aligned on a double-word boundary. The save area words, their displacement from the area origin, and their contents are shown in Figure 43.

WORD	DISPLACEMENT	CONTENTS
1	0	Indicator byte and storage length.
2	4	Address of the save area used by the calling program (stored by the calling program). This save area is in the program that called the calling program.
3	8	Address of the save area in the called program (stored by the called program).
4	12	Retum address (register 14 contents - stored by the called program).
5	16	Entry point address (register 15 contents - stored by the called program).
6	20	Register 0
7	24	Register 1
8	28	Register 2
9	32	Register 3
10	36	Register 4
11	40	Register 5
12	44	Register 6
13	48	Register 7
14	52	Register 8
15	56	Register 9
16	60	Register 10
17	64	Register 11
18	68	Register 12

Figure 43. Save Area Words and Contents in Calling Program

Additional information on the contents of each of the words in a save area is given below:

- Word 1. An indicator byte followed by three bytes that contain the length of allocated storage. Use of these fields is optional, except in programs written in the PL/1 Language.

- Word 2. The address of the save area in the next higher level program. The contents of register 13 must be stored in this word by the calling program (which might be a called program) before the calling program loads register 13 with the address of the current save area.

- Word 3. The address of the save area in the called program, unless the called program is at the lowest level and does not have a save area. (The called program need have a save area only if it is itself a calling program.) Thus the called program, if it contains a save area, stores the save area address in this word. This is done only if save areas are being optionally chained together in descending order.

- Word 4. The return address, which is in register 14 when control is given to the called program. The called program may store the return address in this word if it intends to modify register 14.

- Word 5. The address of the entry point of the called program. This address is in register 15 when control is given to the called program. The called program stores the entry point address in this word if it is required.

- Words 6 through 18. The contents of registers 0 through 12, in that order. The called program stores the register contents in these words if it intends to modify the registers.

SAVE AREA CHAINING

The lowest level program (the current active level) in a chain of program linkages should point to a save area as follows:

- If the lowest level program does not have a save area, register 13 should point to word 1 of the save area in the next higher level program.

- If the lowest level program does have a save area, register 13 should point to word 1 of this save area.

- Whether or not the lowest level program has a save area, register 13 should point to word 1 of the save area in the next higher level program when a RETURN macro instruction is executed.

Note that register 13 need only point to a save area when a linkage occurs.

In all cases, word 2 of each save area should contain a pointer to word 1 of the save area in the next higher level program.

Word 3 of each save area can optionally contain a pointer to word 1 of the save area in the next lower level program.

An example of save area chaining is shown in Figure 44. Two programs (A and B) are shown in the example. Program A has called program B. In this example, program B calls another program which is not shown because it does not call any programs. Programs A and B contain a save area.

REGISTERS (CALLING PROGRAM RESPONSIBILITIES)

The calling program is responsible for the following:

1. Loading register 13 with the address of a save area.

2. Loading register 14 with the return address.

3. Loading register 15 with the entry point address.

4. Loading register 1, if necessary, with the address of a parameter list.

After execution of the calling sequence, the calling program can expect the following to occur as a result of execution of the remainder of the linkages:

1. The contents of registers 2 through 14, the program mask, and the program interruption control area will be unchanged.

2. The contents of registers 0, 1, and 15; the contents of the floating-point registers; and the condition code may be changed.

PROGRAM CONTAINING SAVE AREA	ADDRESS OF SAVE AREA WORD	CONTENTS OF WORD
Program A	A1	Optional
	A2	If A is highest level program, then it contains whatever was assembled there. Otherwise, address of save area in the program that called A.
	A3	B1 (optional)
	A4	Return Address in A (optional)
	A5	Entry point in B (optional)
	A6-A18	Optional
Program B	B1	Optional
	B2	A1
	B3	Not Set
	B4	Return address in B
	B5	Entry point in the lowest level program (no save area).
	B6-B18	Optional

Figure 44. Save Area Chaining

CALL--CALL A PROGRAM

The CALL macro instruction passes control from a program to a specified entry point in another program. The program issuing the CALL macro instruction is referred to as the calling program; the program receiving control is referred to as the called program. The called program must be in main storage when the CALL macro instruction is executed. The called program is brought into main storage in one of two ways:

1. As part of the phase issuing the CALL. In this case, the CALL macro instruction must specify an entry point. When the linkage editor processes a phase containing such a CALL, it includes the called program in the phase.

2. As the phase specified by a LOAD macro instruction. In this case, the CALL macro instruction must specify the program to be called by indicating that the address of its entry point will be loaded into register 15 (the entry point register) before execution of the CALL macro instruction. The LOAD macro instruction must precede the first CALL for the program.

The called program returns control to the calling program by issuing a RETURN macro instruction or its equivalent.

Name	Operation	Operand
[name]	CALL	entry-symbol [,(param-addr,...)]

entry

Specifies the entry point to which control is to be passed. If the symbolic name of an entry point is written, an instruction, L 15,=V(entry), is generated as part of the macro expansion. Control is given to the called program by a branch to the address in register 15 (the entry point register). Entry may be a self-defining value equal to 15 and enclosed in parentheses.

param

Specifies an address (relocatable or absolute expression) to be passed as a parameter to the called program. Terms in the address must not be indexed. The param operands must be written in a sublist, as shown in the format description. If one or more param operands are written, a problem program parameter list is generated. It consists of a full-word for each operand. Each full-word is aligned on a full-word boundary and contains, in its three low-order bytes, the address to be passed. The addresses appear in the parameter list in the same order as in the macro instruction. When the called program is entered, register 1 (the parameter list register) contains the address of the problem program parameter list.

If the entry operand is written as a symbolic name, a V-type address constant is generated by the assembler, and the linkage editor makes the called program part of the calling-program phase. The symbolic name must be either the name of a control section or an assembler language ENTRY statement operand in the called program.

If the entry operand is written as (15), a V-type address constant is not generated. If the called program is not part of the calling-program phase, a LOAD macro instruction must be executed (to bring the program to be called into storage) before the CALL macro instruction is issued.

In the following examples, EX1 gives control to an entry point named ENT. EX2 gives control to an entry point whose address is contained in register 15. Two parameters, ABC and DEF, are passed.

```
EX1  CALL ENT
EX2  CALL (15),(ABC,DEF)
```

A typical macro expansion for the macro instruction NAME CALL SUBR,(P1,P2...,Pn) is:

```
     CNOP 2,4
NAME L    15,=V(SUBR)
     LA   14,*+6+4*n
     BALR 1,15
     DC   A(P1,P2...,Pn)
```

NAME is the symbol in the name field of the macro instruction. n is the number of full-words in the parameter list. SUBR is the symbolic name of the entry point of the called program. P1 through Pn are the addresses to be passed to the called program.

When the CALL macro instruction is executed, it gives control to the called program by branching to the address in register 15.

The (15) entry operand and LOAD macro instruction combination is most useful when the same program is to be called many times during execution of the calling program, but is not needed in main storage throughout execution of the calling program. If the CALL macro instruction is used and a symbolic name written for the entry operand, the called program resides in storage throughout execution of the calling program. This wastes main storage if the called program is not needed during all of the calling-program execution.

SAVE--SAVE REGISTER CONTENTS

The SAVE macro instruction is written at the entry point of a program. Upon entry to the program, SAVE stores the contents of specified registers in a save area provided by the program from which control was given. The saved register contents are reloaded by execution of a RETURN macro instruction.

Name	Operation	Operand
[name]	SAVE	(r1[,r2])

The operands r1, r2 specify the range of the registers to be stored in the save area of the calling program. This area is pointed to by register 13. The operands are written as self-defining values. They should be so written that, when inserted in a STM instruction, they cause desired registers in the range of 14 through 12 (14, 15, 0 through 12) to be stored. Register 14 and 15, if specified, are saved in words 4 and 5 of the save area. Registers 0 through 12, if specified, are saved in words 6 through 18 of the save area. The contents of a given register are always saved in a particular word in the save area. For example, register 3 is always saved in word 9 of the save area, even if register 2 is not saved.

If r2 is omitted, only the register specified by r1 is saved.

RETURN--RETURN TO A PROGRAM

The RETURN macro instruction can reload the registers whose contents were saved by execution of a SAVE macro instruction. After reloading the registers, if any, control returns to the calling program.

Name	Operation	Operand
[name]	RETURN	(r1[,r2])

The operands r1, r2 specify the range of the registers to be reloaded from the save area of the program receiving control. The operands are written as self-defining values. They should be so written that, when inserted in an LM instruction, they cause the loading of registers in the range from 14 through 12 (14, 15, 0 through 12). Registers 14 and 15, if specified, are restored from words 4 and 5 of the save area. Registers 0 through 12, if specified, are restored from words 6 through 18 of the save area. If r2 is omitted, only the register specified by r1 is restored.

The address of the save area must have been loaded into register 13 before execution of this macro instruction.

DASD LABELS

Whenever files of records are written on DASD, each volume must contain standard labels to identify the pack or cell and the logical file(s) on it. When logical IOCS is used for a file, the IOCS routines read, check, and/or write standard labels. When physical IOCS is used, IOCS processes the labels if the DTFPH macro instruction is included in the user's program. The entry TYPEFLE must be specified to indicate whether the file is an input file (read and check labels) or an output file (read and check old labels and write new labels).

The standard labels include one volume label for each pack or cell and one or more file labels for each logical file on the DASD. The following paragraphs describe briefly the organization of labels on disk packs or data cells. Additional information about labels is given in the Data Management Concepts publication, as listed in the abstract of this publication.

Volume Labels

The standard volume label identifies the entire volume and offers volume protection. For systems residence, the volume label is always the third record on cylinder 0, track 0. The first two records on this track of SYSRES are Initial Program Loading (IPL) records. On all other volumes, these records contain binary zeros. The volume-label record consists of a count area, a 4-byte key area, and an 80-byte data area. Both the key area and the first four bytes of the data area contain the label identifier VOL1. The remaining 76 bytes of the data area contain other identifying information such as the volume serial number, and the address of the set of file labels for the pack or cell (see Standard File Labels). The volume label is generally written once, when the DASD is received, by an IBM-supplied utility program.

The standard volume label may be followed by one to seven additional volume labels (starting with record 4 on cylinder 0, track 0). These labels must contain the label identifier VOL2, VOL3, etc in the four-byte key areas and in the first four bytes of the data areas. The other 76 bytes may contain whatever information the user requires. The additional volume labels are also written by the utility program that writes the standard volume label. However, IOCS does not make them available to the user for checking or rewriting when problem programs are executed. These labels are provided for use with Operating System/360 and are always bypassed by the Disk Operating System OPEN routines.

STANDARD FILE LABELS

The standard file labels identify the logical file, give its location(s) on the disk pack or data cell, and offer file protection. The labels for all logical files on a volume are grouped together and stored in a specific area of DASD called the Volume Table of Contents.

The number and format of labels required for any one logical file depends on the file organization (see Standard File Label Formats) and the number of separate areas (extents) of the pack or cell used by the file. The data records for a logical file may be contained within one area of the pack or cell, or they may be scattered in different areas of it. The limits (starting and ending addresses) of each area used by the file are specified by the standard file label(s).

Because each file label contains file limits, the group of labels on the volume is essentially a directory of all files on the volume. Therefore, it is known as the Volume Table of Contents (VTOC). The VTOC itself becomes a file of records (one or more standard-label records per logical file) and, in turn, has a label. The label of the VTOC is the first record in the VTOC. This label identifies the file as the VTOC file, and gives the file limits of the VTOC file. The Volume Table of Contents is contained within one cylinder of a disk pack or data cell. It does not overflow onto another cylinder.

If a logical file of data records is recorded on more than one volume, standard labels for the file must be included in the VTOC of each volume used. The label(s) on each volume identifies the portion of the logical file on the pack or cell and specifies the extent(s) used on it.

STANDARD FILE LABEL FORMATS

All standard file label records have a count area and a 140-byte key/data area. Five standard-label formats are provided.

Format 1. This format is used for all logical files, and it has a 44-byte key area and a 96-byte data area. It is always the first of the series of labels when a file requires more than one label on a disk pack or cell (as discussed in Formats 2 and 3).

The Format-1 label identifies the logical file (by a file name assigned by the user and included in the 44-byte key area), and it contains file- and data-record specifications. It also provides the addresses for three separate DASD areas (extents) for the file. If the file is scattered over more than three separate areas on one pack or cell, a Format-3 label is also required. In this case, the Format-1 label points to the second label set up for the file on this volume.

If a logical file is recorded on more than one volume, a Format-1 label is always created in the VTOC for each volume.

Format 2. This format is required for any file that is organized by the Indexed Sequential File Management System. The 44-byte key area and the 96-byte data area contain specifications unique to this type of file organization.

If an indexed sequential file is recorded on two or more volumes, the Format-2 label is used only on the volume containing the cylinder index. This volume may, or may not, contain data records. The Format-2 label is not repeated on the additional packs (as the Format-1 label is).

Format 3. If a logical file uses more than three extents on any pack or cell, this format is used to specify the addresses of the additional extents. It is used only for extent information. It has a 44-byte key area and a 96-byte data area that provide for 13 extents.

The Format-3 label is pointed to by the Format-1 label for the logical file. In a DTFSD file, it may also be pointed to another Format-3 label. It is included as required on the first pack or cell, or on additional volumes if the logical file is recorded on two or more volumes.

Format 4. The Format-4 label is used to define the VTOC itself. This label is always the first label in the VTOC. This label is also used to provide the location and number of available tracks in the alternate track area.

Format 5. The Format-5 label is used by the Operating System/360 for Direct Access Device Space Management.

USER-STANDARD DASD FILE LABELS

The user may include additional labels to define his file further, if he desires, provided the file is processed sequentially (DTFSR or DTFSD macro specified), by the direct access method (DTFDA macro specified), or by physical IOCS (DTFPH macro specified). User standard file labels are not processed in a file organized and processed by the Indexed Sequential File Management System (DTFIS specified). A file that is to be processed in sequential order (using DTFSR or DTFSD) may have up to eight user header labels and up to eight user trailer labels for a 2311 file, and up to five user header labels and up to five user trailer labels for a 2321 file. The trailer labels can be written to indicate an end-of-volume or end-of-file condition. That is, when the end of an extent on one volume is reached and the next extent is on a different volume, or when the end of the file is reached, user trailer labels can be included to contain whatever trailer information the user desires (for example, a record count for the completed volume).

User-standard labels are not stored in the Volume Table of Contents. Instead, they are written on the first track of the first extent allotted for the logical-file data records. In this case, the user's data records start with the second track in the extent, regardless of whether the labels require a full track. If a file is written on two or more packs or cells, the additional labels are written on each of the packs or cells.

All user-standard labels must be eighty bytes long, and they must contain standard information in the first four bytes. The remaining 76 bytes may contain whatever information the user wants.

The standard information in the first four bytes is used as a record key when reading or writing header labels. The header labels are identified by UHL1, UHL2,...,UHL8. The trailer labels, when applicable, are identified in the key field by UTL0, UTL1,...,UTL7 (or UTL4) although the first four bytes of the labels will contain UTL1-UTL8 (or UTL5). Each user-label set (header or trailer) is terminated by an end-of-file record (a record with data length 0), which is written by IOCS.

For example, if a file has five header labels and four trailer labels, the contents of the user-label track are:

```
R0    Standard information
R1    UHL1--user's 1st header label
R2    UHL2--user's 2nd header label
R3    UHL3--user's 3rd header label
R4    UHL4--user's 4th header label
R5    UHL5--user's 5th header label
R6    UHL6--end-of-file record
R7    UTL1--user's 1st trailer label
R8    UTL2--user's 2nd trailer label
R9    UTL3--user's 3rd trailer label
R10   UTL4--user's 4th trailer
R11   UTL5--end-of-file record
```

If only header labels are used, the user-label track contains:

```
R0    Standard information
R1    UHL1--user's 1st header label
R2    UHL2--user's 2nd header label
        .
        .
        .
R(n)  UHL(n)--user's nth header label where
      n is ≤ 8
R(n+1) UHL(n+1)--end-of-file record
R(n+2) UTL0--end-of-file record
```

The user's label routine can determine if a label is a header or trailer label by testing the first four bytes of the label (see OPEN Macro).

STANDARD TAPE LABELS

When a tape input or output file that has standard labels is opened, IOCS can handle the label checking (on input) or writing (on output). When logical IOCS macros are used in the program, the entry FILABL=STD must be included to specify IOCS processing of labels. When physical IOCS macros are used, the DTFPH entry TYPEFLE must be included to indicate whether this is an input file (check labels) or an output file (write labels).

The standard labels for a tape file are: a volume label, a file header label, and a file trailer label. The volume label, which is the first record (eighty characters) on a reel of tape, identifies the entire volume (reel) and offers volume protection. It contains the label identifier VOL1 in the first four positions, and other identifying information such as the volume serial number. This is a unique number generally assigned to the reel when it is first received in the installation. The volume label is generally written once, when the reel of tape is received, by an IBM-supplied utility program. The standard volume label may be followed by a maximum of seven additional volume labels if desired. These must be identified by VOL2, VOL3, etc in the first four positions of each succeeding label. However, IOCS does not permit the checking or writing of additional volume labels by the user in the problem program. These labels are available for use with the Operating System/360 and are always bypassed on the input for the Disk Operating System.

The volume label set is followed by a standard file header label. This label (eighty characters) identifies the logical file record on the tape and offers file protection. It contains the label identifier HDR1 in the first four positions, and other identifying information such as file identifier, file serial number, creation date, etc. An input tape may contain standard header labels HDR1-HDR8. IOCS checks label HDR1 and bypasses HDR2-HDR8 (these labels are provided for use with the Operating System/360).

The standard file header labels may be followed by a maximum of eight user-written standard labels if desired. If so, the file header labels must be identified by UHL1, UHL2, etc. Labels UHL1-UHL8 may be processed if the DTF entry LABADDR is specified. A tape mark follows the last file header label.

A standard file trailer label is located at the end of a logical file (EOF), or at the end of a volume (EOV) if the logical file continues on another volume. The trailer label has the same format as the header label. It is identified by EOF1 or EOV1 (instead of HDR1) and contains a physical record count (block count). Like the file header label, the standard file trailer label may be followed by user standard trailer labels. These must be identified by UTL1, UTL2, etc.

All user-written standard labels must be eighty characters long and must contain the standard identification in the first four positions. The remaining 76 positions may contain whatever information the user wants. Additional information about tape labels is given in the Data Management Concepts publication, as listed in the abstract of this manual.

If an input tape contains standard labels but the user does not want IOCS to check them, FILABL=NSTD should be specified in the file definition. LABADDR must not be specified in the DTF. A tapemark must immediately follow the label set.

Note: On 7-track tape, standard labels are written on the same density as the data on the tape. All information on a tape reel must be written in single density. These standard labels are written with even parity in the translation mode.

NONSTANDARD TAPE LABELS

Any tape labels that do not conform to the standard-label specifications are considered nonstandard and, if desired, must be read, checked, or written by the user. On input the nonstandard labels may, or may not, be followed by a tapemark. This choice, combined with the user's requirements to check the labels, or not, result in the following four possible conditions that can be encountered:

1. One or more labels, followed by a tapemark, are to be checked.

2. One or more labels, not followed by a tapemark, are to be checked.

3. One or more labels, followed by a tapemark, are not to be checked.

4. One or more labels, not followed by a tapemark, are not to be checked.

For conditions 1 and 2, the DTFSR or DTFMT entries FILABL=NSTD and LABADDR=Name must be specified in the file definition. For condition 3, the entry FILABL=NSTD must be specified. LABADDR is omitted and IOCS skips all labels, passes the tapemark, and positions the tape at the first data record to be read. For condition 4, the entries FILABL=NSTD and LABADDR=Name must be specified. In this case IOCS cannot distinguish labels from data records because there is no tapemark to indicate the end of the labels. Therefore, the user must read all labels even though checking is not desired. This positions the tape at the first data record.

For output files created by logical IOCS, a tapemark may follow the last nonstandard label.

UNLABELED TAPE FILES

On input, unlabeled tapes (FILABL=NO) may or may not contain a tapemark as the first record. If the tapemark is present, the next record is considered to be the first data record. If there is no tapemark, IOCS reads the first record, determines that it is not a tapemark, and backspaces to the beginning of the first record that it considers to be the first data record. For unlabeled output files (FILABL=NO) created by logical IOCS, the first record may be a tapemark.

CTLCHR=ASA

A control character must appear in each logical record if the ASA option is chosen. If the control character for the printer is not valid, a message is given and the job is canceled. If the control character for the card punch is not V or W, the card is selected into pocket 1. The codes are as follows:

Code	Interpretation
(blank)	Space one line before printing
0	Space two lines before printing
-	Space three lines before printing
+	Suppress space before printing
1	Skip to channel 1 before printing
2	Skip to channel 2 before printing
3	Skip to channel 3 before printing
4	Skip to channel 4 before printing
5	Skip to channel 5 before printing
6	Skip to channel 6 before printing
7	Skip to channel 7 before printing
8	Skip to channel 8 before printing
9	Skip to channel 9 before printing
A	Skip to channel 10 before printing
B	Skip to channel 11 before printing
C	Skip to channel 12 before printing
V	Select stacker 1
W	Select stacker 2

CTLCHR=YES

The control character is the command-code portion of the System/360 Channel Command Word used in printing a line or spacing the forms. If the character is not one of the following characters, unpredictable events will occur.

8-Bit Code	Punch Combination	Function
Stacker Selection on 1442		
10000001	12,0,1	Select into stacker 1
11000001	12,1	Select into stacker 2
Pocket Selection on 2540		
00000001	12,9,1	Select into pocket 1
01000001	12,0,9,1	Select into pocket 2
10000001	12,0,1	Select into pocket 3
Stacker Selection on 2520		
00000001	12,9,1	Select into stacker 1
01000001	12,0,9,1	Select into stacker 2
Printer Control		
00000001	12,9,1	Write (no automatic space)
00001001	12,9,8,1	Write and space 1 line after printing
00010001	11,9,1	Write and space 2 lines after printing
00011001	11,9,8,1	Write and space 3 lines after printing
10001001	12,0,9	Write and skip to channel 1 after printing
10010001	12,11,1	Write and skip to channel 2 after printing
10011001	12,11,9	Write and skip to channel 3 after printing
10100001	11,0,1	Write and skip to channel 4 after printing
10101001	11,0,9	Write and skip to channel 5 after printing
10110001	12,11,0,1	Write and skip to channel 6 after printing
10111001	12,11,0,9	Write and skip to channel 7 after printing
11000001	12,1	Write and skip to channel 8 after printing
11001001	12,9	Write and skip to channel 9 after printing
11010001	11,1	Write and skip to channel 10 after printing

8-Bit Code	Punch Combination	Function
11011001	11,9	Write and skip to channel 11 after printing
11100001	11,0,9,1	Write and skip to channel 12 after printing
00001011	12,9,8,3	Space 1 line immediately
00010011	11,9,3	Space 2 lines immediately
00011011	11,9,8,3	Space 3 lines immediately
10001011	12,0,8,3	Skip to channel 1 immediately
10010011	12,11,3	Skip to channel 2 immediately
10011011	12,11,8,3	Skip to channel 3 immediately
10100011	11,0,3	Skip to channel 4 immediately
10101011	11,0,8,3	Skip to channel 5 immediately
10110011	12,11,0,3	Skip to channel 6 immediately
10111011	12,11,0,8,3	Skip to channel 7 immediately
11000011	12,3	Skip to channel 8 immediately
11001011	12,0,9,8,3	Skip to channel 9 immediately
11010011	11,3	Skip to channel 10 immediately
11011011	12,11,9,8,3	Skip to channel 11 immediately
11100011	0,3	Skip to channel 12 immediately
00000011	12,9,3	No op

APPENDIX C: ASSEMBLING THE PROBLEM PROGRAM, DTF'S, AND LOGIC MODULES

All the programs described in this appendix perform the same function, namely, a card-to-disk operation with the following equipment and options:

1. Card reader: 2540 (SYS004).
2. Disk: 2311 with user labels.
3. Record size: 80 bytes.
4. Block size: 408 bytes including 8-byte count field (blocking factor of 5).
5. One I/O area and work area for the card reader.
6. Two I/O areas for the disk.

The following five methods may be used to furnish the DTF's and IOCS logic modules to the card-to-disk program.

1. DTF's, IOCS logic modules, and problem program assembled together.
2. Logic modules assembled separately.
3. DTF's and logic modules assembled separately, label exit, EOF exit, and I/O areas assembled with DTF's.
4. Same as in item 3 except that I/O areas are moved back into main program.
5. Same as in item 4 except that label exit and EOF exit are also moved back into main program.

An example of each of the five preceding methods of assembling the main program, modules, DTF's, and related functions is given in this appendix. In the illustrations that accompany the examples, each dashed arrow represents a symbolic linkage, with an external reference at the base of the arrow, and a label or section definition designating the same symbol at the head of the arrow.

At the points where an arrow is marked with a circle, it will be the programmer's responsibility to define an ENTRY or EXTRN symbol, as applicable.

Each dotted arrow represents a direct linkage. Components are represented by the small rectangles. Assemblies are represented by the larger bordered areas.

The examples are followed by a comparison of the advantages of the five methods.

ASSEMBLING THE PROBLEM PROGRAM, DTF'S, AND LOGIC MODULES TOGETHER (EXAMPLE 1)

Figure 45 illustrates the assembly of the DTF's, logic modules, and problem program.

The assembly source deck is as follows:

```
                                                        Col. 72

CDTODISK    START   0
            BALR    12,0                        Initialize base register.
            USING   *,12                        Establish addressability.
            LA      13,SAVEAREA                 Use reg 13 as pointer to save area.
            OPEN    CARDS,DISK                  Open both files.
NEXT        GET     CARDS,(2)                   Read one card and move it
            PUT     DISK                        to the disk output buffer.
            B       NEXT                        Return for next card.
SAVEAREA    DS      9D                          Save area is 72-byte, double-word
                                                aligned.

EOFCD       CLOSE   CARDS,DISK                  At card-reader EOF, close
            EOJ                                 both files and exit to Job
                                                Control.

MYLABELS    .                                   User's label-processing
            .                                   routine.
            LBRET   2                           Return to main program.

CARDS       DTFCD                           X
                    DEVADDR=SYS004,         X
                    EOFADDR=EOFCD,          X
                    IOAREA1=A1,             X
                    WORKA=YES

DISK        DTFSD                           X
                    BLKSIZE=408,            X
                    IOAREA1=A2,             X
                    IOAREA2=A3,             X
                    IOREG=(2),              X
                    LABADDR=MYLABELS,       X
                    RECFORM=FIXBLK,         X
                    RECSIZE=80,             X
                    TYPEFLE=OUTPUT

A1          DS      80C                         Card-input buffer
A2          DS      408C                        First disk buffer
A3          DS      408C                        Second disk buffer

(1)         CDMOD                           X
                    DEVICE=2540,            X
                    TYPEFLE=INPUT,          X
                    WORKA=YES

            SDMODFO

            END     CDTODISK                    Program-start address
```

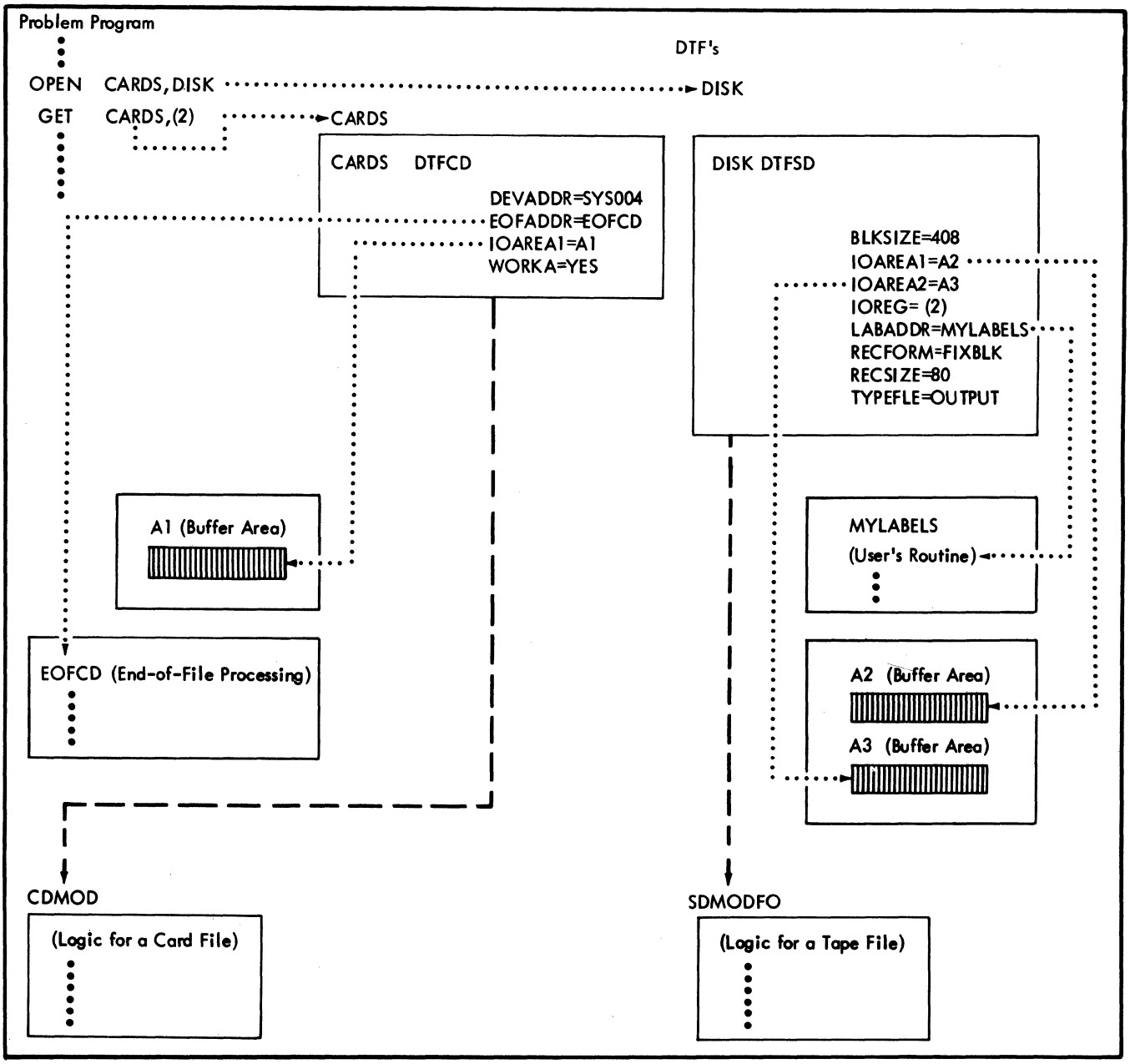


Figure 45. Assembling the Problem Program, DTF's and Modules Together (Example 1)

Assembling the Logic Modules Separately (Example 2)

The main-program source deck is identical to that in Example 1 until (1); at this point, the user simply furnishes the END card. Figure 46 shows the separation of the I/O logic modules.

The two logic modules are assembled as follows:

Col. 72

Card logic module

```
CDMOD                                  X
      DEVICE=2540,                     X
      SEPASMB=YES,                     X
      TYPEFLE=INPUT,                   X
      WORKA=YES
END
```

Disk logic module

```
SDMODFO                                X
      SEPASMB=YES
END
```

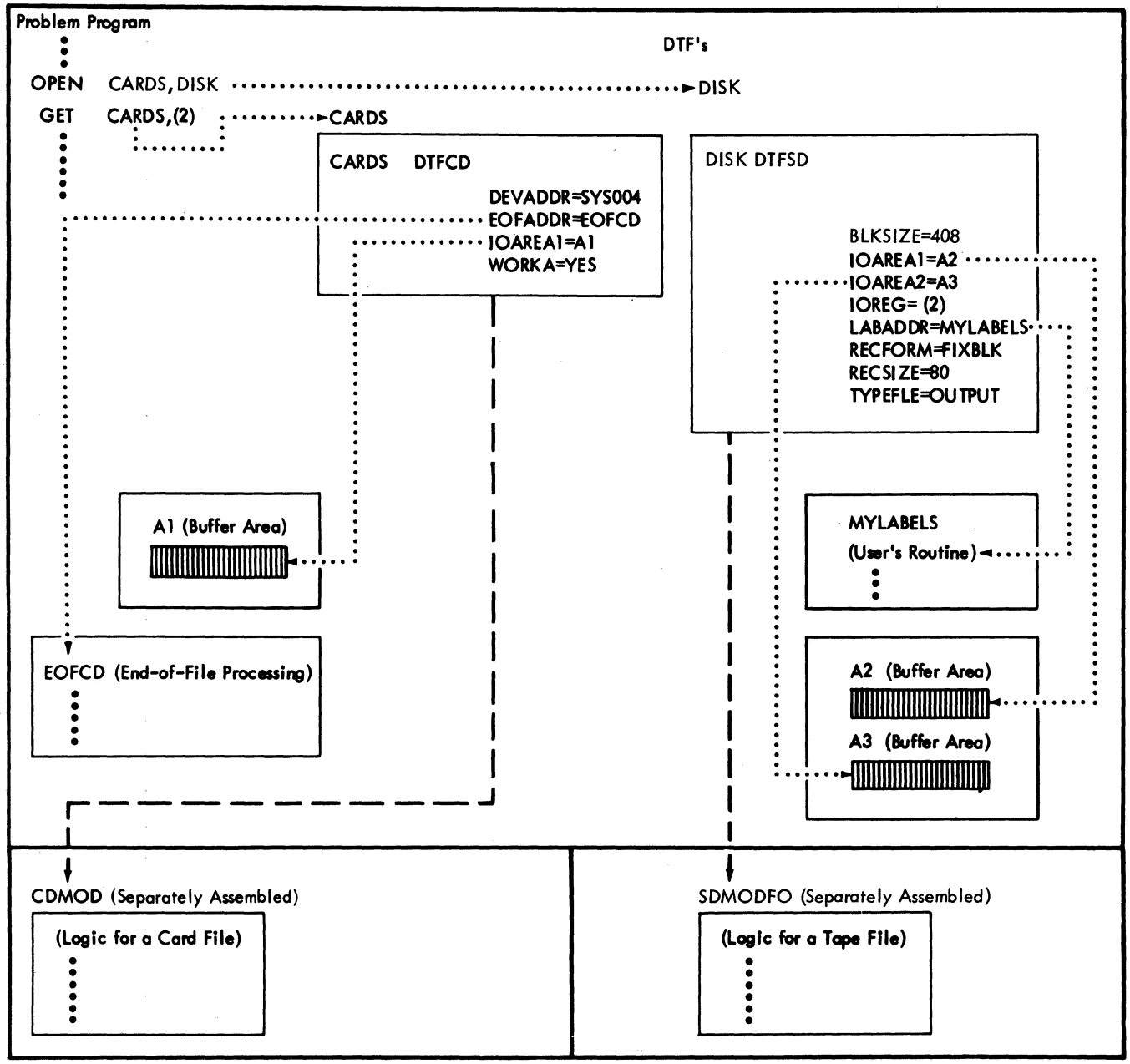


Figure 46. Logic Modules Assembled Separately (Example 2)

After assembly, each logic module is preceded by the appropriate CATALR card; the modules may be added to the system relocatable library during a maintenance run. Thereafter, they are automatically included in the user program by the linkage editor while it prepares the preceding main program for execution.

ASSEMBLING THE DTF'S AND LOGIC MODULES SEPARATELY (EXAMPLE 3)

The main program is assembled as follows:

```
      CDTODISK  START   0
                BALR    12,0
                USING   *,12
                LA      13,SAVEAREA
                OPEN    CARDS,DISK
      NEXT      GET     CARDS,(2)
                PUT     DISK
                B       NEXT
      SAVEAREA  DS      9D

(2)             EXTRN   CARDS,DISK
                END     CDTODISK
```

The logic modules are assembled as in Example 2. Figure 47 shows the separation of the DTF's and logic modules.

The card-file macro instruction and related functions are assembled as follows:

```
                                          Col. 72
      CARDS   DTFCD                          X
                    DEVADDR=SYS004,          X
                    SEPASMB=YES,             X
                    EOFADDR=EOFCD,           X
                    IOAREA1=A1,              X
                    WORKA=YES
              USING *,14

      EOFCD   CLOSE CARDS,DISK
              EOJ

              EXTRN DISK

(3)   A1      DS    80C
              END
```

The disk-file macro instruction and related functions are assembled as follows:

```
                                          Col. 72
      DISK      DTFSD                        X
                      BLKSIZE=408,           X
                      SEPASMB=YES,           X
                      .
                      .
                      .
                      TYPEFLE=OUTPUT
      MYLABELS  BALR  10,0
                USING *,10
                .
                .
                LBRET 2

(4)   A2        DS    408C
(5)   A3        DS    408C
                END
```

In both the card-file and the disk-file assembly above, a USING statement was added because certain user routines are segregated from the main program and moved into the DTF assembly.

When user routines, such as error, label processing, or EOF routines, are segregated from the main program, it is necessary to establish addressability for these routines. The user can provide this addressability by assigning and initializing a base register. In the special case of the EOF routine, the addressability is established by logical IOCS in register 14. For error exits and label processing routines, however, this addressability is not supplied by logical IOCS. Therefore, if the user segregates his error routines, it will be his responsibility to establish addressability for them.

To illustrate how the coding of Example 3, shown above, would look when assembled, Figure 48 contains the printer output.

Notice that in Figure 48 the standard name for the logic modules has been generated: statement 13 of the DTFCD--V(IJCFZIWO), and statement 12 of the DTFSD--V(IJGFOZZZ). These module names appear in the External Symbol Dictionary of each of the respective logic module assemblies.

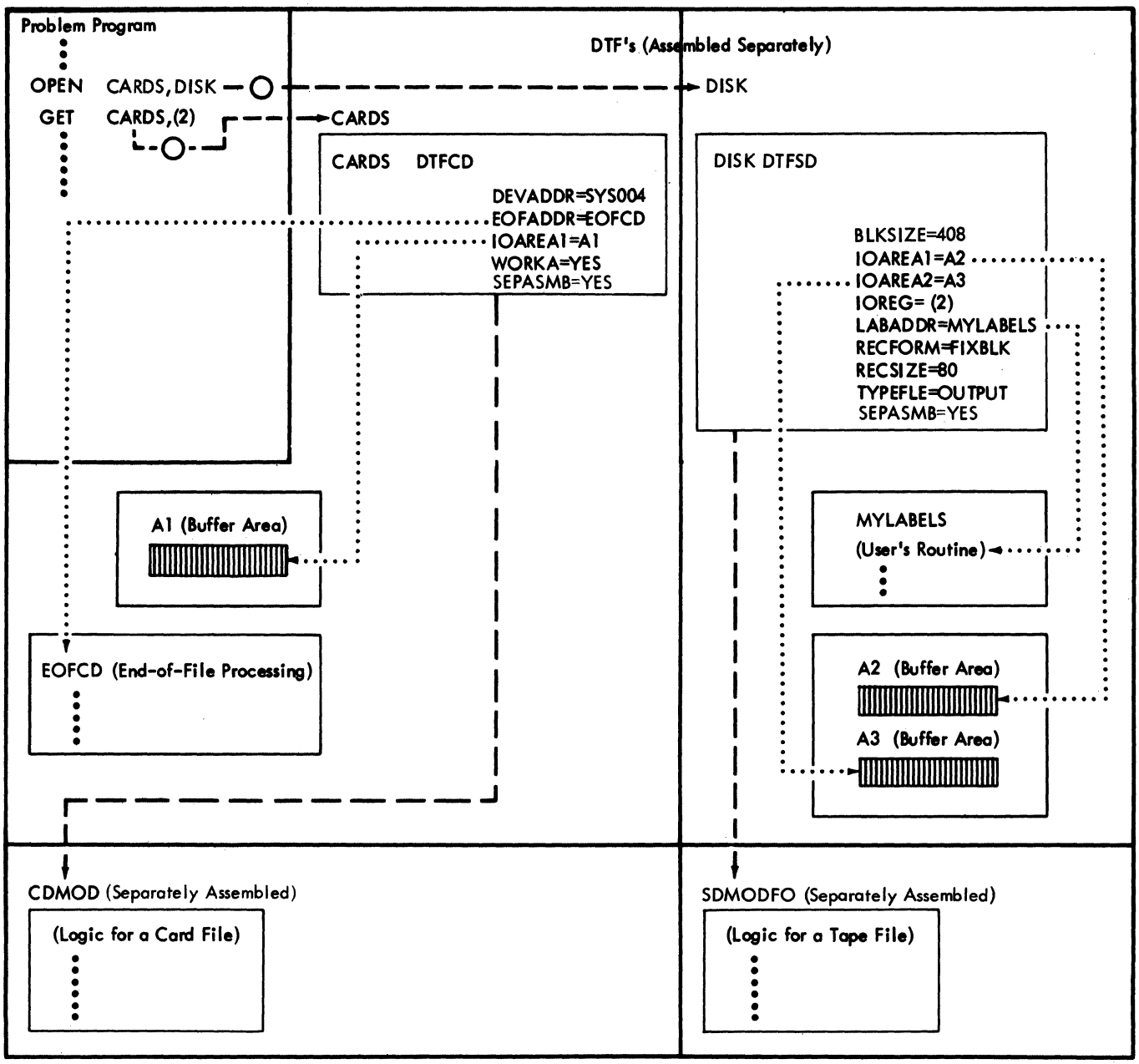


Figure 47. Logic Modules and DTF's Assembled Separately (Example 3)

```
MAIN PROGRAM
                                        EXTERNAL SYMBOL DICTIONARY                                  PAGE    1
 SYMBOL   TYPE ID   ADDR   LENGTH LD ID

CDTODISK   SD   01 000000 000090
CARDS      ER   02
DISK       ER   03
```

Section definition. Control section defined by START statement.
External reference. } Defined by EXTRN statement.
External reference.

```
                                        EXAMPLE 3                                                   PAGE    1

  LOC  OBJECT CODE     ADDR1 ADDR2  STMT   SOURCE STATEMENT                                DD20OCT66 10/26/66

000000                                 1 CDTODISK START 0
000000 05C0                            2          BALR  12,0               INITIALIZE BASE REGISTER
000002                                 3          USING *,12               ESTABLISH ADDRESSABILITY
000002 41D0 C036             00038     4          LA    13,SAVEAREA        USE REGISTER 13 AS POINTER TO SAVE
                                       5          OPEN  CARDS,DISK         OPEN BOTH FILES
                                       6+** SYSTEM CONTROL AND BASIC IOCS 360N-CL-453 CHANGE LEVEL 2-0
000006 0700                            7+         CNOP  0,4
000008                                 8+         DC    0F'0'
000008 4110 C07E             00080     9+         LA    1,=C'$$BOPEN '
00000C 4500 C016             00018    10+IJJ00001 BAL   0,*+4+4*(3-1)
000010 00000000                       11+         DC    A(CARDS)
000014 00000000                       12+         DC    A(DISK)
000018 0A02                           13+         SVC   2
                                      14 NEXT     GET   CARDS,(2)          READ ONE CARD, MOVE TO WORK AREA
                                      15+** CHANGE LEVEL 2-0
00001A 5810 C086             00088    16+NEXT     L     1,=A(CARDS) GET DTF TABLE ADDRESS
00001E 1802                           17+         LR    0,2 GET WORK AREA ADDRESS
000020 58F1 0010             00010    18+         L     15,16(1) GET LOGIC MODULE ADDRESS
000024 45EF 0008             00008    19+         BAL   14,8(15) BRANCH TO GET ROUTINE
                                      20          PUT   DISK               WRITE ON DISK
                                      21+** CHANGE LEVEL 2-0
000028 5810 C08A             0008C    22+         L     1,=A(DISK) GET DTF TABLE ADDRESS
00002C 58F1 0010             00010    23+         L     15,16(1) GET LOGIC MODULE ADDRESS
000030 45EF 000C             0000C    24+         BAL   14,12(15) BRANCH TO PUT ROUTINE
000034 47F0 C018             0001A    25          B     NEXT               GO FOR NEXT CARD
000038                                26 SAVEAREA DS    9D                 72-BYTE SAVE AREA
                                      27          EXTRN CARDS,DISK
000000                                28          END   CDTODISK
000080 5B5BC2D6D7C5D540               29                =C'$$BOPEN '
000088 00000000                       30                =A(CARDS)
00008C 00000000                       31                =A(DISK)
```

Figure 48. Separate Assemblies, (Example 3) (Part 1 of 4)

```
DTFCD ASSEMBLY                                                              PAGE    1
                                  EXTERNAL SYMBOL DICTIONARY
 SYMBOL   TYPE ID  ADDR   LENGTH LD ID

CARDSC    SD  01 000000 0000A0       Section definition.
CARDS     LD     000000          01  Label definition (entry point).   } Generated by specifying SEPASMB=YES in DTFCD macro instruction.
IJCFZIWO  ER  02                     External reference.  Corresponds to V-type address constant generated in DTFCD.
DISK      ER  03                     External reference.  Defined by EXTRN statement.
```

```
                                    EXAMPLE 3                                   PAGE    1

 LOC   OBJECT CODE    ADDR1 ADDR2  STMT   SOURCE STATEMENT                    DD200CT66 10/26/66

                                     1 CARDS    DTFCD DEVADDR=SYS004,                          X
                                                      SEPASMB=YES,                           X
                                                      EOFADDR=EOFCD,                         X
                                                      IOAREA1=A1,                            X
                                                      WORKA=YES
                                     2** SYSTEM CONTROL AND BASIC IOCS 360N-CL-453 CHANGE LEVEL 2-0
                                     3+         PUNCH '         CATALR CARDS'
000000                               4+CARDSC   CSECT
                                     5+         ENTRY CARDS
000000                               6+         DC    0D'0'
000000 000080000000                  7+CARDS    DC    X'000080000000' RES. COUNT,COM. BYTES,STATUS BTS
000006 01                            8+         DC    AL1(1) LOGICAL UNIT CLASS
000007 04                            9+         DC    AL1(4) LOGICAL UNIT
000008 00000020                     10+         DC    A(IJCX0001) CCW ADDRESS
00000C 00000000                     11+         DC    4X'00' CCB-ST BYTE,CSW CCW ADDR.
000010 00                           12+         DC    AL1(0)
000011 000000                       13+         DC    VL3(IJCFZIWO) ADDRESS OF LOGIC MODULE
000014 02                           14+         DC    X'02' DTF TYPE (READER)
000015 01                           15+         DC    AL1(1) SWITCHES
000016 02                           16+         DC    AL1(2) NORMAL COMM.CODE
000017 02                           17+         DC    AL1(2) CNTROL COMM.CODE
000018 00000048                     18+         DC    A(A1) ADDR. OF IOAREA1
00001C 00000034                     19+         DC    A(EOFCD) EOF ADDRESS
000020 020000482000005O             20+IJCX0001 CCW   2,A1,X'20',80
000028 4700 0000            00000   21+         NOP   0 LOAD USER POINTER REG.
00002C D24F D000 E000 00000 00000   22+         MVC   0(80,13),0(14) MOVE IOAREA TO WORKA
000032                              23+IJJZ0001 EQU   *
000032                              24          USING *,14                ESTABLISH ADDRESSABILITY
                                    25 EOFCD    CLOSE CARDS,DISK          END OF FILE ADDRESS FOR CARD READER
                                    26** CHANGE LEVEL 2-0
000032 0700                         27+         CNOP  0,4
000034                              28+EOFCD    DC    0F'0'
000034 4110 E066            00098   29+         LA    1,=C'$$BCLOSE'
000038 4500 E012            00044   30+IJJC0002 BAL   0,*+4+4*(3-1)
00003C 00000000                     31+         DC    A(CARDS)
000040 00000000                     32+         DC    A(DISK)
000044 0A02                         33+         SVC   2
                                    34          EOJ
                                    35** CHANGE LEVEL 2-0
000046 0A0E                         36+         SVC   14
                                    37          EXTRN DISK
000048                              38 A1       DS    80C                 CARD I/O AREA
                                    39          END
000098 5B5BC2C3D3D6E2C5             40                =C'$$BCLOSE'
```

Figure 48. Separate Assemblies, (Example 3) (Part 2 of 4)

```
DTFSD ASSEMBLY                          EXTERNAL SYMBOL DICTIONARY                    PAGE    1
 SYMBOL   TYPE ID   ADDR   LENGTH LD ID

DISKC      SD  01 000000 0003D4       Section definition.
DISK       LD     000000         01   Label definition (entry point).   } Generated by specifying SEPASMB=YES in DTFSD macro instruction.
IJGFOZZZ   ER  02                      External reference. Corresponds to V-type address constant generated in DTFSD.
```

```
                                        EXAMPLE 3                                     PAGE    1

 LOC   OBJECT CODE     ADDR1 ADDR2  STMT   SOURCE STATEMENT                          DD20OCT66 10/26/66

                                      1 DISK     DTFSD BLKSIZE=408,                                X
                                                       SEPASMB=YES,                                X
                                                       IOAREA1=A2,                                 X
                                                       IOAREA2=A3,                                 X
                                                       IOREG=(2),                                  X
                                                       LABADDR=MYLABELS,                           X
                                                       RECFORM=FIXBLK,                             X
                                                       RECSIZE=80,                                 X
                                                       TYPEFLE=OUTPUT
                                      2+* CONSECUTIVE DISK PROCESSING IOCS 360N-10-455 CHANGE LEVEL 2-0
                                      3+           PUNCH '          CATALR DISK'
 000000                               4+DISKC      CSECT
                                      5+           ENTRY DISK
 000000                               6+           DC    0D'0'
 000000 000080000000                  7+DISK       DC    X'000080000000' CCB
 000006 FF                            8+           DC    AL1(255) LOGICAL UNIT CLASS
 000007 FF                            9+           DC    AL1(255) LOGICAL UNIT NUMBER
 000008 00000068                     10+           DC    A(IJGC0001) CCB-CCW ADDRESS
 00000C 00000000                     11+           DC    4X'00' CCB-ST BYTE,CSW CCW ADDRESS
 000010 00                           12+           DC    AL1(0)
 000011 000000                       13+           DC    VL3(IJGFOZZZ) LOGIC MODULE ADDRESS
 000014 20                           14+           DC    X'20' DTF TYPE
 000015 49                           15+           DC    AL1(73) OPEN/CLOSE INDICATORS
 000016 C4C9E2D240404040             16+           DC    CL8'DISK' FILENAME
 00001E 000000000000                 17+           DC    6X'00' BCCHHR ADDR OF F1 LABEL IN VTOC
 000024 0000                         18+           DC    2X'00' VOL SEQ NUMBER
 000026 08                           19+           DC    X'08' OPEN COMMUNICATIONS BYTE
 000027 00                           20+           DC    X'00' XTENT SEQ NO OF CURRENT EXTENT
 000028 00                           21+           DC    X'00' XTENT SEQ NO LAST XTENT OPENED
 000029 0000A0                       22+           DC    AL3(MYLABELS) USER'S LABEL ADDRESS
 00002C 000000A4                     23+           DC    A(A2) ADDRESS OF IOAREA
 000030 80000000                     24+           DC    X'80000000' CCHH ADDR OF USER LABEL TRACK
 000034 0000                         25+           DC    2X'00' LOWER HEAD LIMIT
 000036 00000000                     26+           DC    4X'00' XTENT UPPER LIMIT
 00003A 0000                         27+DISKS     DC    2X'00' SEEK ADDRESS-BB
 00003C 0000FF00                     28+           DC    X'0000FF00' SEARCH ADDRESS-CCHH
 000040 00                           29+           DC    X'00' RECORD NUMBER
 000041 00                           30+           DC    X'00' KEY LENGTH
 000042 0190                         31+           DC    H'400' DATA LENGTH
 000044 00000000                     32+           DC    4X'00' CCHH CONTROL FIELD
 000048 06                           33+           DC    AL1(6) R    CONTROL FIELD
 000049 00                           34+           DC    X'00' SWITCHES
 00004A 018F                         35+           DC    H'399' SIZE OF BLOCK-1
 00004C 0000000000                   36+           DC    5X'00' CCHHR BUCKET
 000051 00                           37+           DC    X'00'
 000052 0E29                         38+           DC    H'3625' TRACK CAPACITY CONSTANT
 000054 5821 0058          00058     39+           L     2,88(1) LOAD USER'S IOREG
 000058 000000AC                     40+           DC    A(A2+8) DEBLOCKER-INITIAL POINTER
 00005C 00000050                     41+           DC    F'80' DEBLOCKER-RECORD SIZE
 000060 0000023B                     42+           DC    A(A2+8+400-1) DEBLOCKER-LIMIT
 000064 0A                           43+           DC    AL1(10) LOGICAL INDICATORS
 000065 000000                       44+           DC    AL3(0) USER'S ERROR ROUTINE
 000068 0700003A40000006             45+IJGC0001  CCW   7,*-46,64,6 SEEK
 000070 3100003C40000005             46+           CCW   X'31',*-52,64,5 SEARCH ID EQUAL
 000078 0800007000000000             47+           CCW   8,*-8,0,0 TIC
```

'igure 48. Separate Assemblies, (Example 3) (Part 3 of 4)

DTFSD (Continued)

```
                                        EXAMPLE 3                                                    PAGE    2

 LOC  OBJECT CODE    ADDR1 ADDR2  STMT   SOURCE STATEMENT                                          DD200CT66 10/26/66

000080 1D00023C00000198              48+          CCW   X'1D',A3,0,400+8 WRITE COUNT KEY AND DATA
000088 3100003C40000005              49+          CCW   X'31',DISKS+2,64,5 SEARCH ID EQUAL
000090 0800008800000000              50+          CCW   8,*-8,0,0 TIC
000098 1E00009830000001              51+          CCW   30,*,48,1 VERIFY
0000A0                               52+IJJZ0001 EQU   *
0000A0 05A0                          53 MYLABELS BALR  10,0                 INITIALIZE BASE REGISTER
0000A2                               54          USING *,10                 ESTABLISH ADDRESSABILITY
                                     55 *        .                          USER'S LABEL PROCESSING ROUTINE
                                     56 *        .
                                     57          LBRET 2                    RETURN TO IOCS
                                     58+* CHANGE LEVEL 2-0
0000A2 0A09                          59+         SVC   9 BRANCH BACK TO IOCS
0000A4                               60 A2       DS    408C                 FIRST DISK I/O AREA
00023C                               61 A3       DS    408C                 SECOND DISK I/O AREA
                                     62          END
```

CDMOD ASSEMBLY

```
                                    EXTERNAL SYMBOL DICTIONARY
 SYMBOL   TYPE ID  ADDR   LENGTH LD ID

IJCFZIW0  SD  01 000000 000060  Section definition. CSECT name generated by CDMOD macro instruction.
```

```
                                        EXAMPLE 3

 LOC  OBJECT CODE    ADDR1 ADDR2  STMT   SOURCE STATEMENT

                                      2          PRINT NOGEN
                                      3          CDMOD                                                X
                                                       DEVICE=2540,                                    X
                                                       SEPASMB=YES,                                    X
                                                       TYPEFLE=INPUT,                                  X
                                                       WORKA=YES
                                     73          END
```

SDMODFO ASSEMBLY

```
                                    EXTERNAL SYMBOL DICTIONARY
 SYMBOL   TYPE ID  ADDR   LENGTH LD ID

IJGFOZZZ  SD  01 000000 0001D4  Section definition. CSECT name generated by SDMODFO macro instruction.
```

```
                                        EXAMPLE 3

 LOC  OBJECT CODE    ADDR1 ADDR2  STMT   SOURCE STATEMENT

                                      2          PRINT NOGEN
                                      3          SDMODFO                                              X
                                                       SEPASMB=YES
                                    169          END
```

Figure 48. Separate Assemblies, (Example 3) (Part 4 of 4)

The DTF assembly generates a table that contains no executable code.

Each of the DTF tables is preceded by the appropriate CATALR card. These two object decks can be cataloged into the relocatable library together with the logic modules as follows:

```
// JOB CATRELOC
// EXEC MAINT
   (DTFCD Assembly)
   (DTFSD Assembly)
   (CDMOD Assembly)
   (SDMODFO Assembly)
/*
```

Alternately, the object decks from these assemblies (DTF tables and logic modules) can be furnished to the linkage editor, along with the main program object deck. The sequence is as follows:

```
// JOB CATALOG
// OPTION CATAL
   INCLUDE
   PHASE name,*
   (Object deck, main program)
   (Object deck, DTFCD assembly)
   (Object deck, DTFSD assembly)
   (Object deck, CDMOD assembly)
   (Object deck, SDMODFO assembly)
/*
// EXEC LNKEDT
/&
```

Note: It is not necessary to remove the CATALR card because the linkage editor bypasses it.

DTF'S AND LOGIC MODULES ASSEMBLED SEPARATELY, I/O AREAS WITH MAIN PROGRAM (EXAMPLE 4)

The main program is identical to that of Example 3 except that the following four cards are inserted after the card marked (2):

```
A1      DS      80C
A2      DS      408C
A3      DS      408C
        ENTRY   A1,A2,A3
```

The separate assembly of logic modules is identical to that in Example 3.

In the card-file assembly of Example 3, replace the card marked (3) with the following card:

```
        EXTRN A1
```

Similarly, in the disk-file assembly of the previous example, replace the cards marked (4) and (5) with the following card:

```
        EXTRN A2,A3
```

Figure 49 shows the separation of the logic modules, DTF's and I/O areas.

ASSEMBLING DTF'S AND LOGIC MODULES SEPARATELY: I/O AREAS, LABEL EXIT, AND END-OF-FILE EXIT WITH MAIN PROGRAM (EXAMPLE 5)

In addition to the changes described in the previous example, the label exit and the end-of-file exit may be assembled separately. Figure 50 illustrates these separate assemblies. The main program is assembled as follows:

```
CDTODISK  START   0
          BALR    12,0
          USING   *,12
          LA      13,SAVEAREA
          OPEN    CARDS,DISK
NEXT      GET     CARDS,(2)
          PUT     DISK
          B       NEXT
SAVEAREA  DS      9D

EOFCD     CLOSE   CARDS,DISK
          EOJ

MYLABELS  .
          .
          .
          LBRET   2

          EXTRN   CARDS,DISK
A1        DS      80C
A2        DS      408C
A3        DS      408C
          ENTRY   A1,A2,A3,EOFCD,MYLABELS
          END     CDTODISK
```

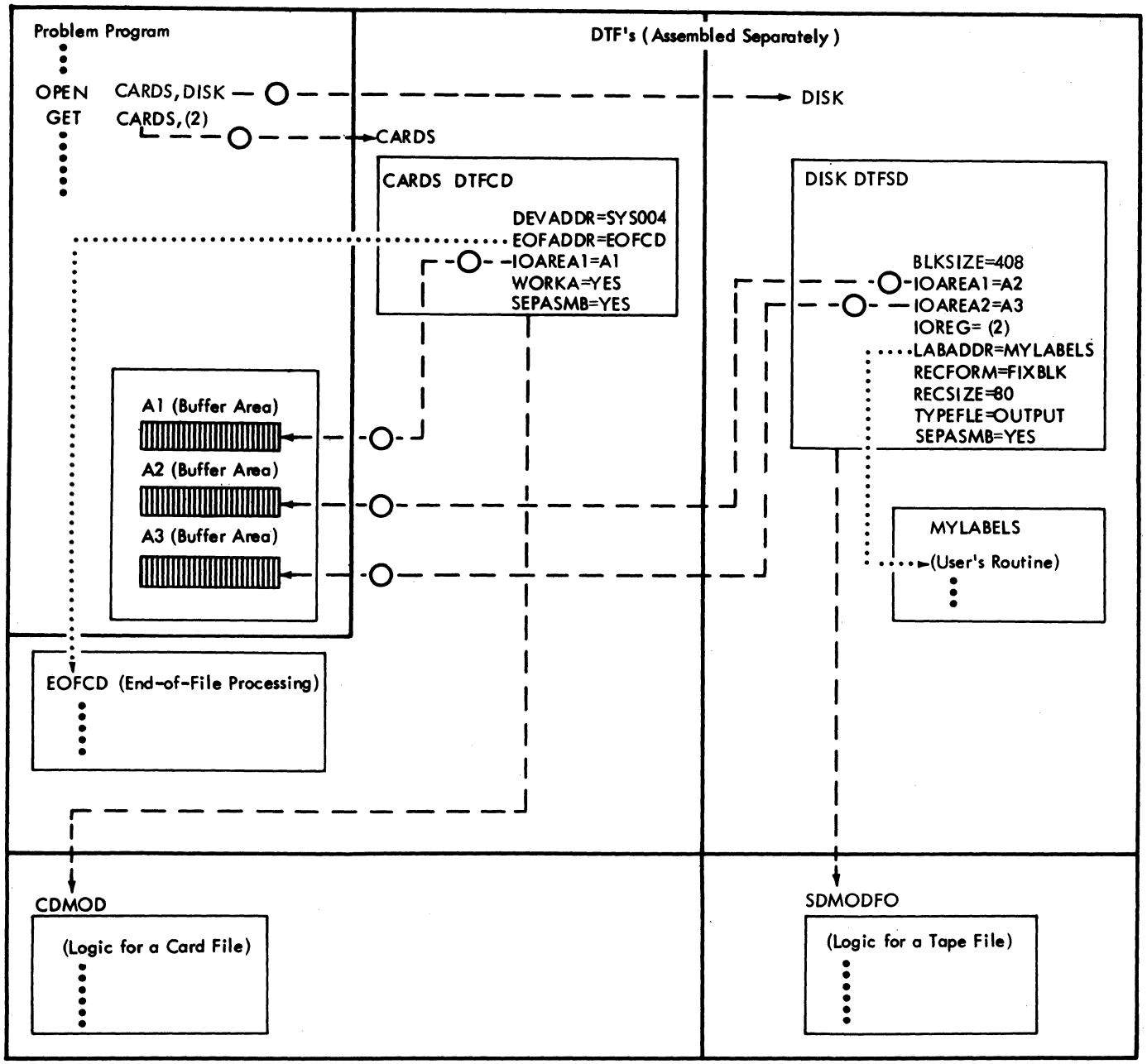


Figure 49. Logic Modules and DTF's Assembled Separately, I/O Areas With Main Program (Example 4)

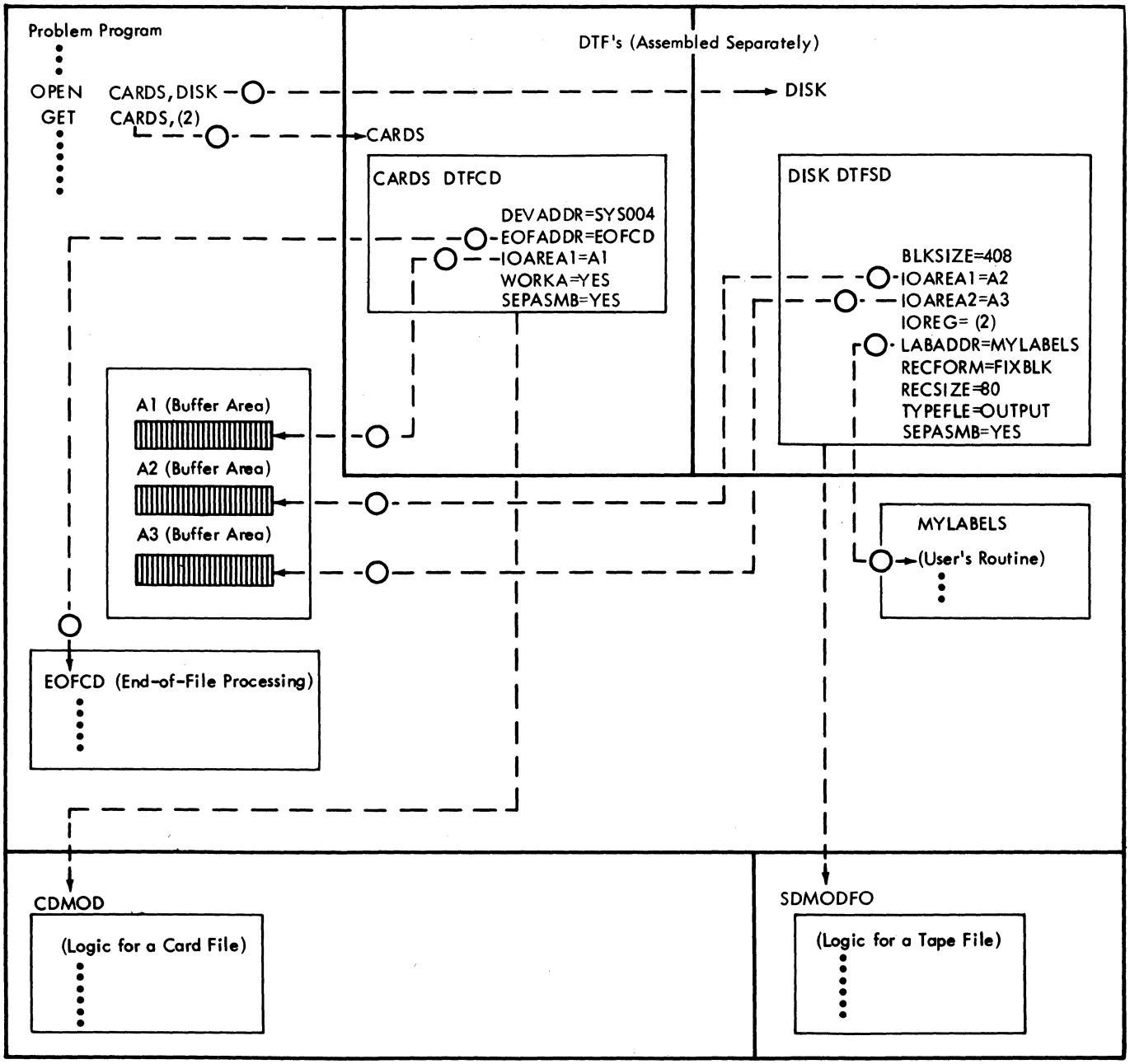


Figure 50. DTF's, and Logic Modules Assembled Separately; I/O Areas Label Exit, EOF Exit with Main Program (Example 5)

The file definition instructions are separately assembled as follows:

```
                                                    Col. 72
CARDS    DTFCD    DEVADDR=SYS004,                      X
                  WORKA=YES,                           X
                  EOFADDR=EOFCD,                       X
                  SEPASMB=YES,                         X
                  IOAREA1=A1
         EXTRN    EOFCD,A1
         END

DISK     DTFSD    BLKSIZE=408,                         X
                  TYPEFLE=OUTPUT,                      X
                  SEPASMB=YES,                         X
                  .
                  .
                  .
                  .
                  IOAREA1=A2,                          X
                  IOAREA2=A3
         EXTRN    A2,A3,MYLABELS
         END
```

The separate assembly of logic modules is identical to that in Examples 3 and 4.

Comparison of the Five Methods

Example 1 requires the most assembly time and the least linkage-edit time. Because the linkage editor is substantially faster than the assembler, frequent reassembly of this program will require more total time for program preparation than Examples 2 through 5.

Example 2 segregates the IOCS logic modules from the remainder of the program. Because these modules are generalized, they can serve several different applications. Thus, they are normally retained in the system relocatable library for ease of access and maintenance.

When a system pack is generated or when it requires maintenance, the IOCS logic modules that are required for all applications should be identified and generated onto it. Each such module requires a separate assembly and a separate catalog operation, as shown in Examples 2 through 5. Many assemblies, however, can be batched together as can many catalog operations.

Object programs produced by COBOL, PL/1, and RPG require one or more IOCS logic modules in each executable program. These modules are usually assembled (as in Example 2) during generation of a system pack and are permanently cataloged into the system relocatable library.

Example 3 shows how a standardized IOCS package can be separated almost totally from a main program. Only the imperative IOCS macro instructions remain: OPEN, CLOSE, GET, and PUT. All file parameters, label processing, other IOCS exits, and buffer areas have been preassembled. If there are few IOCS changes in an application compared to other changes, this method reduces to a minimum the total development/maintenance time. This approach also serves to standardize file descriptions so that they can be shared among several different applications. This reduces the chance of one program creating a file that is improperly accessed by subsequent programs. In Example 3, the user need only be concerned with the record format and the general register pointing to the record. He can virtually ignore the BLKSIZE, LABADDR, etc parameters in his application program, although he must ultimately consider their effect on main storage, job-control cards, etc.

In Example 4, a slight variant of Example 3, the I/O buffer areas are moved into the main program rather than being assembled with the DTF's. In Example 5, the label processing and exit functions are also moved into the main program. Examples 4 and 5 show how buffers and IOCS facilities can be moved between main program and separately assembled modules. If user label processing is standard throughout an installation, label exits should be assembled together with the DTF's. If each application requires special label processing, label exits should be assembled into the main program.

```
                                        EXTERNAL SYMBOL DICTIONARY                                PAGE    1
  SYMBOL   TYPE ID  ADDR   LENGTH LD ID

           PC  01 003000 0004E0
  IJCFZIZ0 ER  02
  IJFFZZZZ ER  03
  IJFFBZZZ ER  04
  IJDFZZZZ ER  05
```

```
TEST   CREATING AND PROCESSING NON-STANDARD LABELS                                                PAGE    1

  LOC  OBJECT CODE    ADDR1 ADDR2  STMT   SOURCE STATEMENT                                    DD200CT66 10/26/66

                                     2          PRINT ON,NOGEN,NODATA                                        NSTD0004
003000                               3          START 12288                                                  NSTD0005
                                     4 *                                                                     NSTD0006
                                     5 READER   DTFCD DEVICE=2540,DEVADDR=SYSIPT,BLKSIZE=80,TYPEFLE=INPUT,  *NSTD0007
                                                EOFADDR=ENDCARD,IOAREA1=IOAREA                               NSTD0008
                                    26 *                                                                     NSTD0009
                                    27 TAPEOUT DTFMT DEVADDR=SYS004,IOAREA1=IOAREA,BLKSIZE=80,TYPEFLE=OUTPUT,*NSTD0010
                                                LABADDR=LABELOUT,READ=FORWARD,FILABL=NSTD                    NSTD0011
                                    58 *                                                                     NSTD0012
                                    59 TAPEIN  DTFMT DEVADDR=SYS004,IOAREA1=IOAREA,BLKSIZE=80,TYPEFLE=INPUT, *NSTD0013
                                                EOFADDR=ENDTAPE,READ=FORWARD,FILABL=NSTD,REWIND=NORWD,       *NSTD0014
                                                LABADDR=LABELIN                                              NSTD0015
                                    93 *                                                                     NSTD0016
                                    94 TAPEIN2 DTFMT DEVADDR=SYS004,IOAREA1=IOAREA,BLKSIZE=80,TYPEFLE=INPUT, *NSTD0017
                                                EOFADDR=ENDTAPE2,READ=BACK,FILABL=NSTD                       NSTD0018
                                   129 *                                                                     NSTD0019
                                   130 PRINT  DTFPR DEVICE=1403,DEVADDR=SYSLST,IOAREA1=IOAREA,BLKSIZE=80     NSTD0020
                                   151 *                                                                     NSTD0021
                                   152 CONSOLE DTFCN BLKSIZE=80,DEVADDR=SYSLOG,IOAREA1=CAREA,RECFORM=FIXUNB, *NSTD0022
                                                WORKA=YES                                                    NSTD0023
                                   204 *                                                                     NSTD0024
                                   205 *                                                                     NSTD0025
003208 0520                        206 START  BALR  2,0                SET UP A BASE REGISTER               NSTD0026
00320A                             207        USING *,2                                                      NSTD0027
                                   208 * ** ROUTINE TO WRITE TAPE                                            NSTD0028
                                   209        OPEN  TAPEOUT            TO WRITE NSTD RECORDS               NSTD0029
                                   217 GETCARD GET  READER             READ A CARD FROM THE CARD READER    NSTD0030
                                   222        PUT   TAPEOUT            WRITE CARD IMAGE ON TAPE            NSTD0031
003232 47F0 2010          0321A    227        B     GETCARD            BRANCH AND GET ANOTHER CARD         NSTD0032
                                   228 ENDCARD CLOSE TAPEOUT           TO WRITE NSTD TRAILER LABEL         NSTD0033
                                   236 * ** ROUTINE TO READ TAPE FORWARD                                     NSTD0034
                                   237        OPEN  PRINT,TAPEIN       TO PROCESS NSTD LABEL               NSTD0035
                                   246 GETTAPE GET  TAPEIN             GET A CARD IMAGE FROM TAPE          NSTD0036
                                   251        PUT   PRINT              PRINT CARD IMAGE ON PRINTER         NSTD0037
003272 47F0 2050          0325A    256        B     GETTAPE            BRANCH AND GET ANOTHER TAPE RECORD  NSTD0038
                                   257 ENDTAPE CLOSE TAPEIN            PROCESS NSTD LABELS                 NSTD0039
                                   265 * ** ROUTINE TO READ TAPE BACKWARDS                                   NSTD0040
                                   266        OPEN  TAPEIN2            BYPASS NSTD LABELS                  NSTD0041
                                   274 GETTAPE2 GET TAPEIN2            READ A TAPE RECORD                  NSTD0042
                                   279        PUT   PRINT              PRINT RECORD                        NSTD0043
0032AE 47F0 208C          03296    284        B     GETTAPE2           BRANCH AND GET ANOTHER TAPE RECORD  NSTD0044
                                   285 ENDTAPE2 CLOSE PRINT,TAPEIN2    BYPASS NSTD RECORDS                 NSTD0045
                                   294        CNTRL TAPEIN2,REW        REWIND TAPE TO LOAD POINT           NSTD0046
                                   300        EOJ                      NORMAL END OF JOB                   NSTD0047
                                   303 * ** LABEL CREATION ROUTINE                                           NSTD0048
```

```
                                                                                                  PAGE    2

  LOC  OBJECT CODE    ADDR1 ADDR2  STMT   SOURCE STATEMENT                                    DD200CT66 10/26/66

0032D6 4900 22D2          034DC    304 LABELOUT CH    0,=X'00D6'          OPEN OR CLOSE                    NSTD0049
0032DA 4770 20F0          032FA    305        BNE   TRAILOUT            BRANCH IF CLOSE                  NSTD0050
0032DE D227 221C 21CC 03426 033D6  306        MVC   IOAREA(40),HEADER   MOVE HEADER TO I/O AREA          NSTD0051
                                   307 RITELAB EXCP  OUTCCB              WRITE LABEL                      NSTD0052
                                   311        WAIT  OUTCCB              WAIT FOR COMPLETION              NSTD0053
                                   317        LBRET 2                   RETURN CONTROL TO IOCS           NSTD0054
0032FA D227 221C 21F4 03426 033FE  320 TRAILOUT MVC  IOAREA(40),TRAILER MOVE TRAILER LABEL TO I/O AREA   NSTD0055
003300 47F0 20DA          032E4    321        B     RITELAB             BRANCH TO WRITE THE LABEL        NSTD0056
                                   322 * ** LABEL PROCESSING ROUTINE                                      NSTD0057
003304 4900 22D2          034DC    323 LABELIN  CH    0,=X'00D6'          OPEN OR CLOSE                    NSTD0058
003308 4780 212C          03336    324        BE    HEADIN              OPEN TIME                        NSTD0059
                                   325 TRAILIN EXCP  INCCB               READ A TRAILER LABEL             NSTD0060
                                   329        WAIT  INCCB               WAIT FOR I/O COMPLETION          NSTD0061
```

```
003320 9101 2270       0347A          335           TM    INCCB+4,X'01'        TEST FOR A TAPE MARK                    NSTD0062
003324 4710 2164              0336E   336           BO    EXITEOF              BRANCH IF YES                           NSTD0063
003328 D527 221C 21F4 03426 033FE     337           CLC   IOAREA(40),TRAILER   COMPARE TRAILER LABEL                  NSTD0064
00332E 4780 2102              0330C   338           BE    TRAILIN              BRANCH TO GET ANOTHER RECORD            NSTD0065
003332 47F0 2152              0335C   339           B     ERRLAB               BRANCH IF LABELS DO NOT CONPARE         NSTD0066
                                      340 HEADIN    EXCP  INCCB                READ A HEADER LABEL                     NSTD0067
                                      344           WAIT  INCCB                WAIT FOR COMPLETION                     NSTD0068
00334A 9101 2270       0347A          350           TM    INCCB+4,X'01'        TEST FOR A TAPE MARK                    NSTD0069
00334E 4710 2168              03372   351           BO    EXIT                 BRANCH IF YES                           NSTD0070
003352 D527 221C 21CC 03426 033D6     352           CLC   IOAREA(40),HEADER    DOES HEADER LABEL COMPARE               NSTD0071
003358 4780 212C              03336   353           BE    HEADIN               IF YES, BRANCH AND READ TAPE            NSTD0072
                                      354 ERRLAB    PUT   CONSOLE,LABELERR     PUT LABEL ERROR MESSAGE                 NSTD0073
                                      360           EOJ                        TERMINATE JOB                           NSTD0074
00336E 4800 22D4              034DE   363 EXITEOF   LH    0,=C'EF'             INDICATE EOF TO IOCS                    NSTD0075
                                      364 EXIT      LBRET 2                    RETURN CONTROL TO IOCS                  NSTD0076
                                      367 * CONSTANTS                                                              NSTD0077
003374 4040404040404040               368 CAREA     DC    CL50' '              CONSOLE I/O AREA                        NSTD0078
0033A6 E4E2C5D940D3C1C2               369 LABELERR  DC    C'USER LABELS DO NOT COMPARE. ABNORMAL END OF JOB.'           NSTD0079
0033D6 E4E2C5D940C8C5C1               370 HEADER    DC    CL40'USER HEADER LABEL'                                      NSTD0080
0033FE E4E2C5D940E3D9C1               371 TRAILER   DC    CL40'USER TRAILER LABEL'                                     NSTD0081
003426 4040404040404040               372 IOAREA    DC    CL80' '              INPUT/OUTPUT AREA                       NSTD0082
                                      373 INCCB     CCB   SYS004,INCCW         READ TAPE CCB                           NSTD0083
                                      384 OUTCCB    CCB   SYS004,OUTCCW        WRITE TAPE CCB                          NSTD0084
003496 0000
003498 0200342600000028               395 INCCW     CCW   X'02',IOAREA,X'00',40   READ TAPE CCW                     NSTD0085
0034A0 0100342600000028               396 OUTCCW    CCW   X'01',IOAREA,X'00',40   WRITE TAPE CCW                    NSTD0086
003208                                397           END   START                                                       NSTD0087
0034A8 5B5BC2D6D7C5D540               398                 =C'$$BOPEN '
0034B0 5B5BC2C3D3D6E2C5               399                 =C'$$BCLOSE'
0034B8 00003000                       400                 =A(READER)
0034BC 00003038                       401                 =A(TAPEOUT)
0034C0 00003090                       402                 =A(TAPEIN)
```

TEST CREATING AND PROCESSING NON-STANDARD LABELS PAGE 3

```
  LOC  OBJECT CODE    ADDR1 ADDR2  STMT   SOURCE STATEMENT                    DD200CT66 10/26/66
0034C4 00003150                       403                 =A(PRINT)
0034C8 000030F0                       404                 =A(TAPEIN2)
0034CC 00003486                       405                 =A(OUTCCB)
0034D0 00003476                       406                 =A(INCCB)
0034D4 00003180                       407                 =A(CONSOLE)
0034D8 000033A6                       408                 =A(LABELERR)
0034DC 00D6                           409                 =X'00D6'
0034DE C5C6                           410                 =C'EF'
```

Notes: 1. IOCS wrote the first tapemark because the TAPEMARK=NO parameter was omitted.
2. IOCS always writes the tapemark following the data.
3. IOCS wrote the two tapemarks after the user trailer label.

Notes: 1. IOCS reads the first tapemark or bypasses it if user labels are not checked.
2. Upon encountering the second tapemark IOCS branches to the user label address.
3. After the user reads the third tapemark he should issue a LBRET 1 and IOCS will branch to the end-of-file address.