SPECTRA 70

RADIO CORPORATION OF AMERICA • ELECTRONIC DATA PROCESSING

TOS/TDOS FCP-EXECUTIVE COMMUNICATION MACROS REFERENCE MANUAL

RCA

TAPE OPERATING SYSTEM (TOS)

TAPE-DISC OPERATING SYSTEM (TDOS)

FILE CONTROL PROCESSOR and EXECUTIVE COMMUNICATION MACROS REFERENCE MANUAL

RCA

SPECTRA 70

RADIO CORPORATION OF AMERICA • ELECTRONIC DATA PROCESSING

TAPE OPERATING SYSTEM (TOS)

TAPE-DISC OPERATING SYSTEM (TDOS)

FILE CONTROL PROCESSOR
and
EXECUTIVE COMMUNICATION MACROS
REFERENCE MANUAL

RADIO CORPORATION OF AMERICA

70-00-608

DECEMBER 1967

The information contained herein is subject to change without notice. Revisions may be issued to advise of such changes and/or additions.

First Printing: June 1966
Reprinted: August 1966
Reissued: December 1966
Reprinted: August 1967

This manual has been reprinted to include all revisions previously released through December 1967.

FOREWORD

♦ When using the POS/TOS/TDOS Assembly Language the following publications are applicable:

1. Assembly System Reference Manual (POS/TOS/TDOS) (No. 70-00-602)

   This publication contains the specifications for writing the basic source language and macro definition language of the Spectra 70 Assembly System. These language specifications are applicable to the Primary Operating System, Tape Operating System, and Tape-Disc Operating System.

2. POS File Control Processor and Supervisor Communication Macros (No. 70-00-605)

   This publication contains descriptions of the logical and physical FCP for the Primary Operating System (70/25-35-45-55). In addition, the Supervisor Communication Macros are described.

3. TOS/TDOS File Control Processor and Executive Communication Macros (No. 70-00-608)

   This publication contains descriptions of the logical and physical FCP for the Tape Operating System and Tape Disc Operating System (70/35-45-55). In addition, the Executive Communication Macros are described. In this publication all references to TOS also apply to TDOS.

4. DOS File Control Processor and Executive Communication Macros (Not available)

   This publication contains descriptions of the logical and physical FCP for the Disc Operating System (70/45-55). In addition, the Executive Communication Macros are described.

The organization of these publications is such that a user may combine the required documentation to fit his needs. For example, a POS FCP user would require the Assembly System and POS File Control Processor and Supervisor Communication Macros publications whereas a TOS/TDOS FCP user would require the Assembly System and TOS/TDOS File Control Processor and Executive Communication Macros publications.

CONTENTS

CONTENTS
(Cont'd)

*TDOS Only

CONTENTS
(Cont'd)

*TDOS Only

1. FILE CONTROL PROCESSOR

INTRODUCTION

General

♦ The File Control Processor (FCP) is a generalized Input/Output Programming System that controls input/output functions in conjunction with the Executive Control System. It provides a set of related routines that simplifies control of input and output devices and automatically processes the simultaneous I/O capabilities of the Spectra 70/35-45-55 Processors. Program requests for input/output operations are made in the form of macro instructions.

The FCP routines are designed in a modular fashion so only those routines called for, based on file definitions, are included in the bound object program. During an assembly when FCP encounters a macro instruction, it verifies the macro, generates the required coding and constants, and determines the proper FCP routine(s) required to service that macro. The output of the assembly process is a program object module file. This object module file must then be processed through the link-bind routine (TOS Linkage Editor) to produce the machine language object program. If I/O control macros (action macros) are assembled apart from the DTF description macros, the FCP generates external references (EXTRN's) for each of the I/O control macros. During the link-bind operation, these external references are satisfied.

The File Control Processor provides the programmer with two levels of control in specifying his input/output requirements. The primary level is called logical level FCP, and the secondary level is called physical level FCP.

Logical Level FCP

♦ The logical level FCP controls those functions required to locate a logical record for processing and provides for automatic handling of labels, error recovery, record batching, record transfer, and reel swapping. Under this concept, the programmer always works at the logical record level and need not be concerned with the physical reading and writing of files. Logical FCP uses physical FCP to execute an I/O command whenever it determines that a transfer of data is required. The program communicates with logical FCP by using two types of macro instructions: File Definition macros and I/O Control macros.

File Definition Macros

♦ File Definition macros perform the following functions:

1. Describe characteristics of the logical file.
2. Describe the physical device on which the logical file resides.
3. Identify options to be taken under certain predefined conditions.
4. Contain addresses of the program's written routines that perform certain options.

I/O Control Macros

- I/O Control macros perform the following functions:

  1. Make files available for processing (including label checking).
  2. Make logical records available for processing (blocking and deblocking).
  3. Alternate I/O areas (when two I/O areas are used).
  4. Store logical records after processing.
  5. Perform control operations such as rewind and paper advance.
  6. Handle end-of-reel and end-of-file conditions (including label writing).
  7. Make files unavailable for processing.

Physical Level FCP

- The physical level FCP is used when the program has special I/O requirements that make it undesirable to use the logical level FCP. Physical level FCP consists of two types of macro instructions: File Definition macros and I/O Control macros. A File Definition macro is required only when the program desires the FCP to perform standard label checking or when writing with reel-swapping is required. No other physical level files need be defined. I/O Control macros are provided to permit control of any I/O device in the system. When using physical level FCP, the program must perform the physical processing of the file as well as provide its own logical data processing routines. This includes record blocking or deblocking, data movement to work areas, initiation of error recovery, and I/O interrupt analysis. To assist users of physical level FCP, a limited set of logical level I/O control macros is provided (OPEN, CLOSE, LBRET, and FEOV).

General Considerations

- 1. All input and output files must conform to published Spectra 70 label and format standards.
  2. Read and write operations are performed with automatic blocking and deblocking of fixed- or variable-length records. When all the records in an input block have been processed, logical FCP automatically reads in a new batch of records. When an output area has been filled, logical FCP automatically writes the block of records to the output device.
  3. Four general registers are used by the FCP; General Registers 0 and 1 are used to pass parameters between the program and FCP, and General Registers 14 and 15 are used for entry and reentry points (see Macro Expansion section for register usage).
  4. The use of alternate I/O areas for simultaneously processing in one area and reading into or writing out of another area, is provided in logical FCP.
  5. For tapes containing more than one file, FCP provides for positioning of the tape to the proper file.
  6. When multireel tape files are mounted on two tape stations, FCP provides for processing of one reel simultaneously with rewind-unload on the alternate reel.

General Considerations *(Cont'd)*

7. Because the Translate and Test instruction uses General Register 1, caution must be used when running a program with FCP. If this instruction is used, General Register 1 must be stored prior to use and restored after use.
8. User programs containing literals must include a LTORG statement to insure that the origin of the literals is in the user program section rather than in the FCP generated coding, which is a separate CSECT containing a LTORG statement.
9. The START (Start Assembly) instruction in the program must have a valid nonblank name if logical level FCP is used in the program.

DEVICES CONTROLLED BY FCP

◆ The peripheral devices controlled by FCP are listed in table 1-1.

Table 1-1. Devices Controlled by Logical FCP

Model No.	Device
70/221-10,-11,-21	Paper Tape Reader/Punch
70/234-10,-11	Card Punch
70/236-10,-11	Card Punch
70/237-10,-21,-22	Card Reader
70/242-10,-20	Printer, Medium Speed
70/243-10,-20	Printer, High Speed
70/248-10	Bill Feed Printer (Continuous Forms only)
70/432-1,-2	Magnetic Tape Unit
70/442-1,-2	Magnetic Tape Unit
70/445-1,-2	Magnetic Tape Station
70/564	Disc Storage Unit
70/565-12,-13	Drum Memory Unit
70/568-1,-2	Mass Storage Unit

Device Error Recovery

◆ Error recovery procedures are provided for each I/O device or class of I/O devices and are used at the program's option.

If the program is performing I/O operations at the logical FCP level, error recovery procedures are performed automatically. If the error persists after such error recovery, a program option is taken to ignore the error, bypass the record, exit to a program-defined routine or terminate the job.

If the program is performing I/O operations at the physical device control level, it must initiate error recovery procedures. (Error recovery options are determined by the user flag byte in the CCB.)

LABEL PROCESSING

General

♦ To verify proper processing of data files, the FCP provides a system of label creation and label checking. Labels are defined as records that appear outside the limits of data in a file, and that are used for file identification and control. Standard formats have been designed for beginning and ending labels, and the FCP automatically checks and creates standard labels for those files defined as having standard labels. FCP allows the user to bypass most errors. Tape label standards conform to published Spectra 70 System Standards for standard label formats. In general, the FCP routine for standard tape labels performs the following functions:

Header Labels on Input Tape

1. Verify proper volume.
2. Verify proper identification.
3. Issue appropriate error messages.

Header Labels on Output Tape

1. Analyze old header label to check expiration date.
2. Write new header label.
3. Issue appropriate error messages.

Trailer Labels on Input Tape

1. Verify input block count with trailer block count field.
2. Analyze the label identifier field for EOF and EOV and take appropriate action.

Trailer Labels on Output Tape

1. Write trailer labels with output block count and an EOF identifier if an end-of-file condition exists, or EOV identifier if an end-of-volume condition exists.

Label Creation and Checking

Standard Tape Labels

♦ *Volume Labels* - To create or replace a volume label on a tape, Tape Volume Initializer routine is used. This is usually done when a reel is placed in service at an installation.

Note: FCP routines do not create, alter, or update volume labels.

Standard Tape Labels (Cont'd)

File Labels - When checking a volume and a header label on either an input or an output tape, a VOL card and a file label card TPLAB or the DTF entry LABNAME must be used. If the VOL and TPLAB label cards are used, each card is read by the Executive Control System as a run-time parameter and stored in memory for use by the label checking routines in the OPEN and CLOSE macros. A VOL and TPLAB card must be submitted for each file that has standard labels to be checked. In addition, a FILES parameter card may be used for file positioning. If label run-time parameter cards are not desired, the DTF entry LABNAME can be specified. This entry indicates the area that contains the label information required to perform label processing.

For input files, the OPEN macro checks the standard header label against information supplied by the label cards or contained in the area specified by the DTF entry LABNAME. For output files, the expiration date of the output tape is first checked to ensure that it does not contain an active file.

The CLOSE macro deactivates processing on input files, and writes the trailer label on output files. This trailer label is either an EOF (end-of-file) label or an EOV (end-of-volume) label, whichever is appropriate.

Nonstandard Tape Labels

♦ Tape labels that do not conform to the standard label specifications are considered nonstandard and must be read, checked, or written by the program. Since input and output data may or may not be preceded by a tape mark, the following four conditions are possible:

1. Nonstandard labels, followed by a tape mark, to be checked on input or written to output.
2. Nonstandard labels, not followed by a tape mark, to be checked or written.
3. Nonstandard labels, followed by a tape mark, which are not to be checked.
4. Nonstandard labels, not followed by a tape mark, which are not to be checked.

Beginning Labels

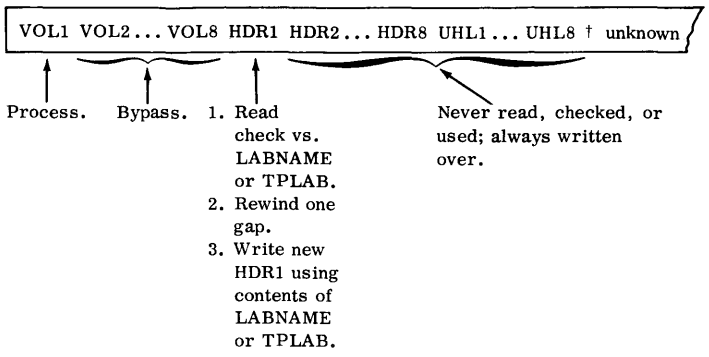
For conditions 1 and 2, the DTFSR entries must specify nonstandard labels and the address of a user's routine to do the reading and writing. The address of the user's routine to read or write nonstandard labels is the LABADDR entry in the DTFSR. In condition 1, the user's program must position beyond the tape mark. For condition 3, nonstandard labels are specified but a user's routine is not necessary because FCP positions the tape at the first data record to be read. For condition 4, nonstandard labels and a program routine address are specified. The program has to read the label to position the tape because there are no tape marks to indicate the end of labels.

Ending Labels

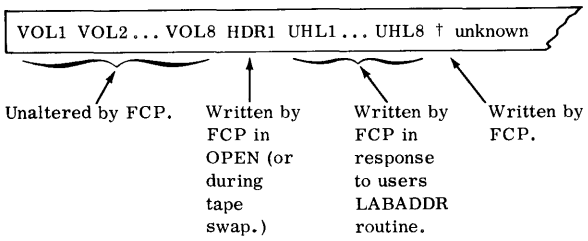
After the user program has read or written its nonstandard labels following the tape mark that follows the data records, the program must indicate to FCP whether and end-of-file or an end-of-volume condition exists.

Standard Labels - Multireel or Single Reel File - OUTPUT - OPEN and Tape Swap

◆ Contents of Output Tape Before the OPEN:

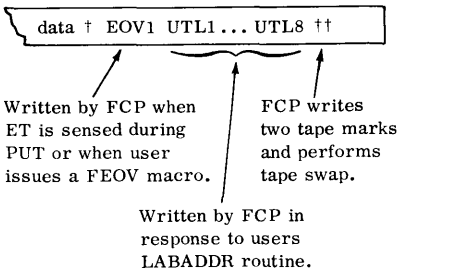


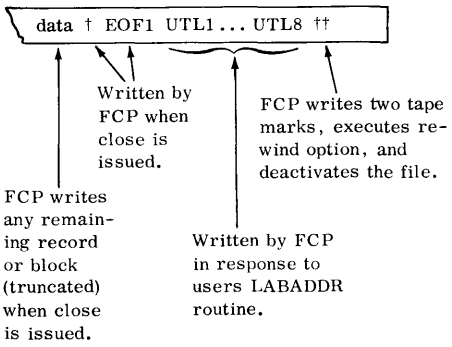
Contents of the output tape after the OPEN:



Legend: Each VOL represents a volume label.
Each HDR represents a file header label.
Each UHL represents a user header label.
† represents a tape mark.

Standard Labels Multireel or Single Reel Files - OUTPUT - CLOSE, and Tape Swap

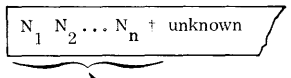




Nonstandard Labels - Multireel or Single Reel File - OUTPUT - OPEN, CLOSE, and Tape Swap

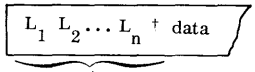
◆ All volumes:

Contents of the output tape before the OPEN:



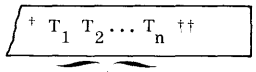
Action by FCP: Never read, checked, or used; always written over.

Contents after the OPEN:



Written by user at physical level.

Contents after the CLOSE:



Written by user at physical level.

Legend

N = old nonstandard label.
T = new nonstandard trailer labels.
L = new nonstandard header label.

On seven-level tapes, labels must be written in even parity, with pack/unpack Off, and translate On (code 48,88, or C8). The density must be the same as the density of the data.

Example:

C8 - labels, F0 - data.

FCP Label Treatment for Random Access Devices

◆ The VOL1 label is located on track 0 of cylinder 0 of the volume. For disc and drum the VOL1 label is always record 3 (R3) on this track. For mass storage it is always Record 1 (R1), on this track. The VOL1 random access label is identical in format to the standard VOL1 tape label, except for field 5, which contains the location of the volume table of contents. For disc and drum there can be from one to seven additional VOLn labels following the VOL1 label. VOLn labels occupy the same track as the VOL1 label. Additional VOLn labels are not allowed for mass storage.

Corresponding to the standard tape HDR1 file label is the random access format 1 record for the file. Each file on a random access volume has one format 1 record. It is located in the volume table of contents. There are no HDRn labels in random access FCP.

User header labels, (UHLn) and user trailer labels (UTLn) are allowed for the disc and drum but not for the mass storage unit. All UHLn and UTLn labels for a file are located on the first track of the first extent for the file.

The VOL1 label is created by the Initializer routine. A dummy format 1 record is also created by the Initializer. The format 1 record for a file is updated by the Allocator routine. UHLn and UTLn are written for output files by FCP when the user exits from his DTF entry LABADDR routine.

Files with nonstandard labels and unlabeled files are not allowed in random access FCP.

FCP Label Treatment for Random Access Devices *(Cont'd)*

The first track of the first extent of a file is automatically reserved by logical level FCP for UHLn and UTLn labels whether the user has specified their use or not. Therefore, if processing serially the user should start his data records on the second track of his first extent.

FILE PROCESSING

General

◆ Files must be made available by issuing an OPEN macro. This macro verifies that the correct tape is mounted (if standard labels are used) and also positions the file for processing.

The program can issue an OPEN macro anywhere in the program and, under certain circumstances, it can also reopen a file after it has been closed.

Overlay

◆ In order to use memory more effectively, FCP overlays the routines employed in the OPENing of files, the CLOSing of files, and the end-of-reel logic associated with a file. Complete control over the incorporation of these routines into a program and their use during the execution of the object program are maintained by FCP. The program has no responsibility nor requirement in order to implement this overlay scheme.

RECORD PROCESSING

General

◆ FCP processes the following different types of record formats:

1. *Fixed-Length Unblocked* - These records are all the same length as the physical record.

2. *Fixed-Length Blocked* - These blocks can contain two or more logical records. The number of records in each block is normally constant except for the last block in a file, which can be shorter. However, the programmer can write short blocks by using the TRUNC macro. The size of these truncated blocks must be a multiple of the length of the logical records. Record padding to fill out the block is not necessary for these truncated blocks.

3. *Variable-Length Unblocked* - These records have different lengths and each one contains a field that indicates its overall size. Each physical record contains one logical record and this record contains a field that indicates its individual size.

4. *Variable-Length Blocked* - These records have different lengths, and two or more logical records form one physical record. Each block of records contains a field that indicates its overall size, and each record within the block contains a field that indicates its individual size. The blocking factor for this type of record can be variable.

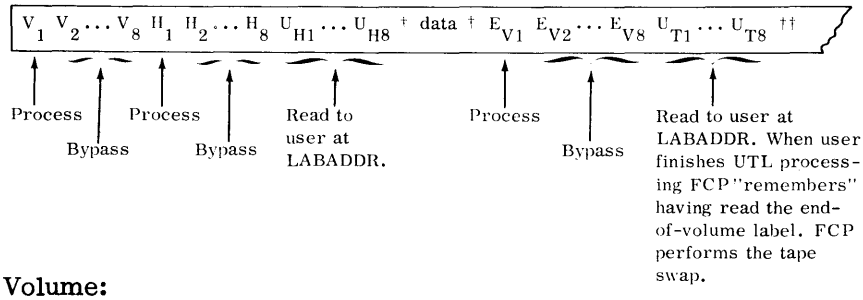
Unlabeled Tape Files

◆ Unlabeled tape files are assumed to contain tape marks (9-channel tape files) as the first and last records. The user has the option of not having the initial tape mark. On output files FCP writes a tape mark preceding the first data block unless the user has specified the DTFSR entry TPMARK = NO. On input files FCP positions the tape past the initial tape mark unless the user has specified the DTFSR entry TPMARK = NO. For input and output, if TPMARK = NO has been specified, FCP makes the file available for processing. The terminal tape mark is always required on unlabeled tape files.

Tape Label Handling by FCP

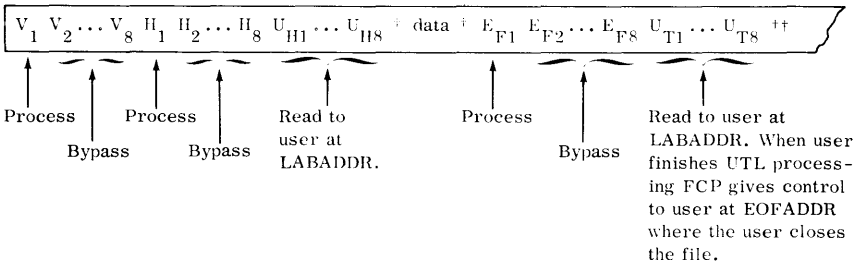
Standard Labels - Multireel or Single Reel File - INPUT - OPEN, CLOSE, and Tape Swap

◆ First and Intermediate Volumes:



Action by FCP:

Last Volume:



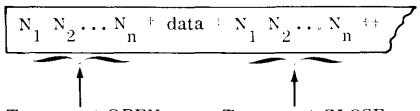
Action by FCP:

Legend

V - volume label.	E_V - end-of-volume trailer label.
H - header label.	E_F - end-of-file trailer label.
U_H - user header label.	† - represents a tape mark.
U_T - user trailer label.	

Nonstandard Labels - Multireel File - INPUT - OPEN, CLOSE, and Tape Swap

◆ All Volumes:



Action by FCP:

To user at OPEN.

User reads the nonstandard labels at physical level, stops with LBRET macro.

To user at CLOSE.

User reads the nonstandard labels at physical level. User returns with LBRET, tells FCP whether this is end-of-volume or end-of-file.

If end-of-volume FCP performs tape swap. If end-of-file FCP branches to user's EOFADDR routine.

Legend: Each N represents a nonstandard label.
† represents a tape mark.

General *(Cont'd)*

5. *Undefined* – These records do not conform to any of the previous four formats. In this format, FCP controls only the actual reading and writing of the records. Logical processing must be performed in the user's program.

All record formats must conform to published Spectra 70 System Standards.

Simultaneity

♦ The user's program uses the I/O Control component of the Executive and the logical FCP routines to read and store records in memory. These routines provide for overlapping the physical transfer of data during processing. The amount of overlapping or simultaneity actually achieved is governed by the user's program through the assignment of I/O areas and work area.

An I/O area is the area of memory to or from which a block of data is physically transferred. A work area is an area used for processing an individual record from the block of data.

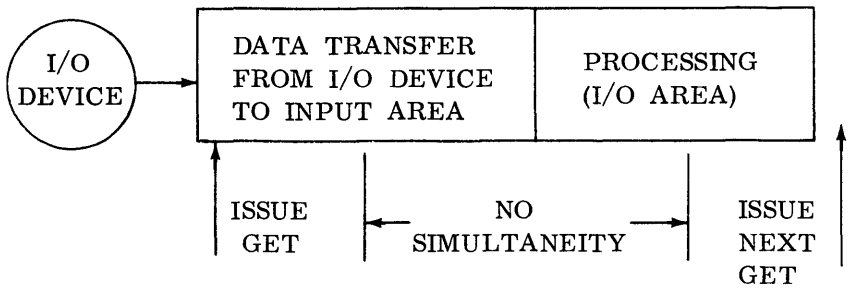
The following combinations of I/O areas and work areas are possible:

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

In addition, certain devices are buffered, increasing the amount of processing I/O overlap.

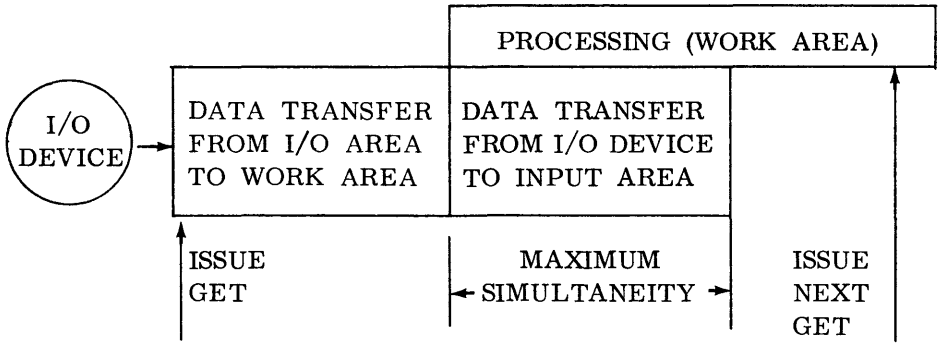
The following diagrams illustrate the amount of simultaneity that can be achieved using certain combinations of I/O areas and work areas. (See table 1-2.) All diagrams are shown using magnetic tape devices as input.

1. *Unblocked Records, One I/O Area, No Work Area*



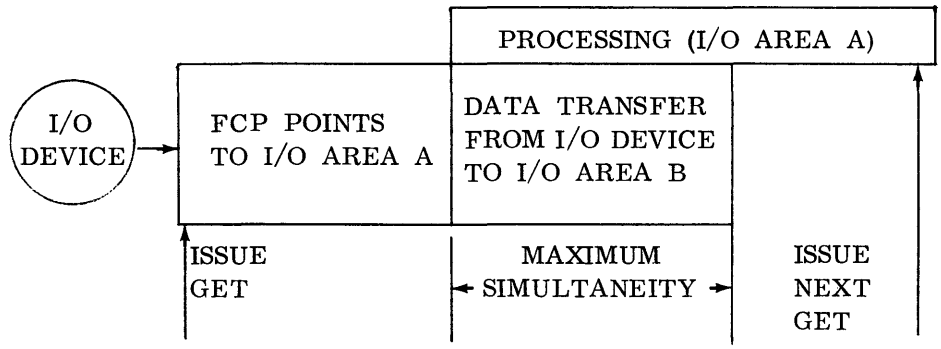
Simultaneity *(Cont'd)*

2. *Unblocked Record, One I/O Area, Work Area*



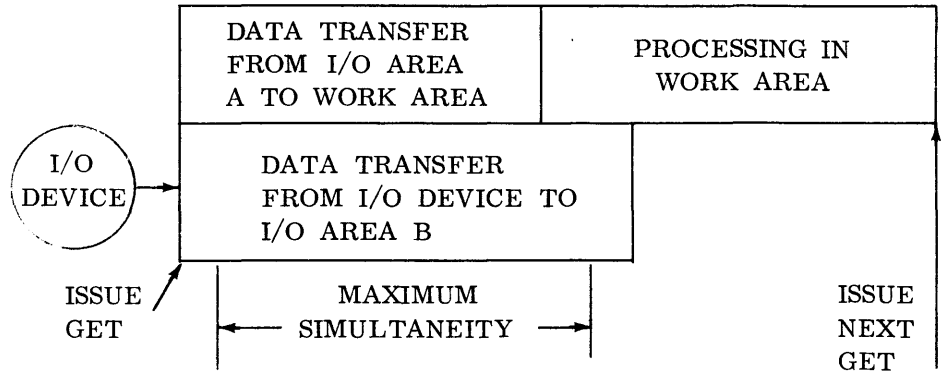
When the GET is issued, data from the previous GET is transferred to the work area; the next record is then read into the I/O area as the previous record is being processed in the work area. A record move to the work area is performed by FCP.

3. *Unblocked Records, Two I/O Areas, No Work Area*



When the GET is issued, FCP points to I/O area A; while the next record is being read into I/O area B, the previous record is being processed in I/O area A. No record move is required by FCP.

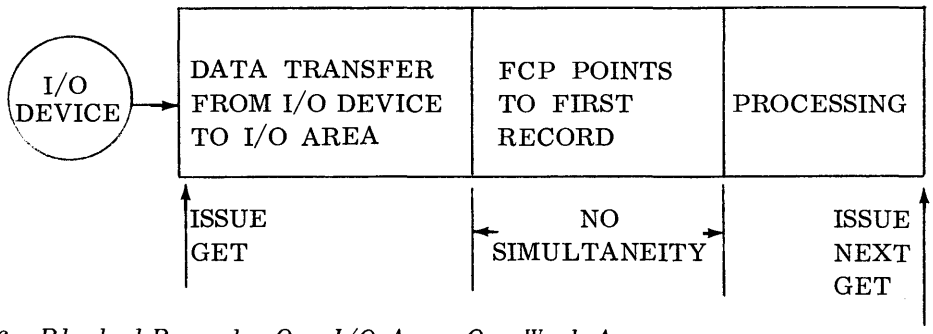
4. *Unblocked Records, Two I/O Areas, Work Area*



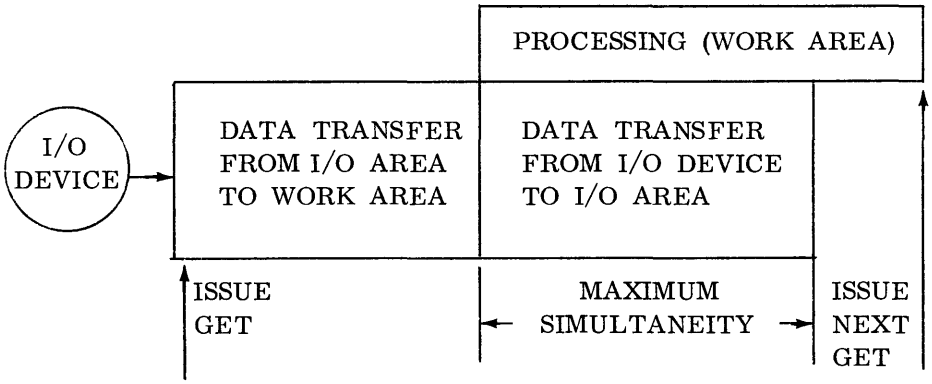
Simultaneity *(Cont'd)*

There is no advantage to having a work area with two I/O areas because maximum simultaneity is still the data transfer time between the I/O device and the I/O area. A work area also requires more memory.

5. *Blocked Records, One I/O Area, No Work Area*

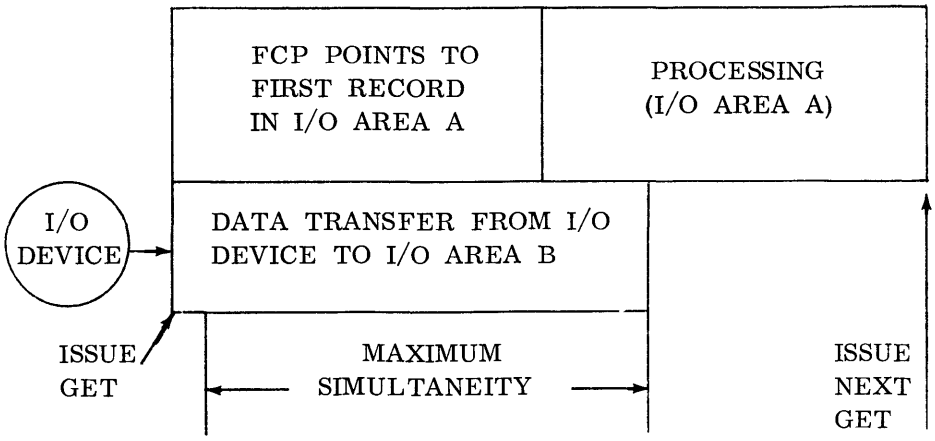


6. *Blocked Records, One I/O Area, One Work Area*



The above diagram shows the record accessed as being the last record on the block. The GET for all other records in the block involves only a data transfer between the work area and the I/O area with no simultaneity involved.

7. *Blocked Records, Two I/O Areas, No Work Area*



DYNAMIC ALTERNATE TRACK ASSIGNMENT

◆ Dynamic Alternate Track Assignment (DALTA) provides the linkage of alternate tracks to defective tracks discovered at object or execution time, thus reducing the probability of run failure because of recording media failure. It is an extension of the error analysis performed by TOS/TDOS FCP for random access devices (70/564, 70/565, and 70/568). Note that this function is not available for DTFPH users since the input/output requests for DTFPH are issued by the problem program rather than by the File Control Processor.

The supporting routine, Random Access Volume Initializer, must be used to preformat the volumes. The Volume Table of Contents contains the location of the alternate track area. The programmer must allocate a sufficient number of alternate tracks at initialization time.

The Dynamic Alternate Track Assignment routine performs the following:

1. Reads the standard volume label.
2. Reads the format 4 label of the VTOC.
3. Searches the alternate track area for an unassigned track.
4. Copies the prime track to the alternate track and inserts the unrecordable record.
5. Flags the defective prime track and links it to the newly assigned alternate track.

The routine does not perform if any of the following conditions occurs:

1. More than one unrecoverable read error during the unload of the prime track.
2. Home address cannot be written on the prime track.
3. The track descriptor record cannot be written on the prime track.
4. An alternate track is not available.
5. Home address cannot be written on the alternate track.
6. The record overflow feature is used (OVERFLO = YES) specified in the DTF).

The following conditions are assumed if DALTA is to perform.

1. Use of standard volume labels.
2. Standard use of home address and track descriptor records.
3. The only unreadable record on the track is the one on which the write failed.
4. CAPREC = YES has been specified in the DTF.
5. VERIFY = YES has been specified in the DTF for 70/564 and 70/565. This entry is not necessary for the 70/568.

DYNAMIC ALTERNATE TRACK ASSIGNMENT (Cont'd)

◆ To generate the DALTA routine the programmer must specify in his DTFDA or DTFSR the keyword parameter ALTAR=YES. DALTA routine must not be used if the file specifies OVERFLOW=YES, or if UPDATE=YES was used on a file that contains records which were written with the physical overflow feature employed.

Alternate Track Address

◆ A four-byte area has been established starting at FILENAME+48, The purpose of this area is for storage of the CCHH address of the alternate track assigned or used by error recovery or DALTA.

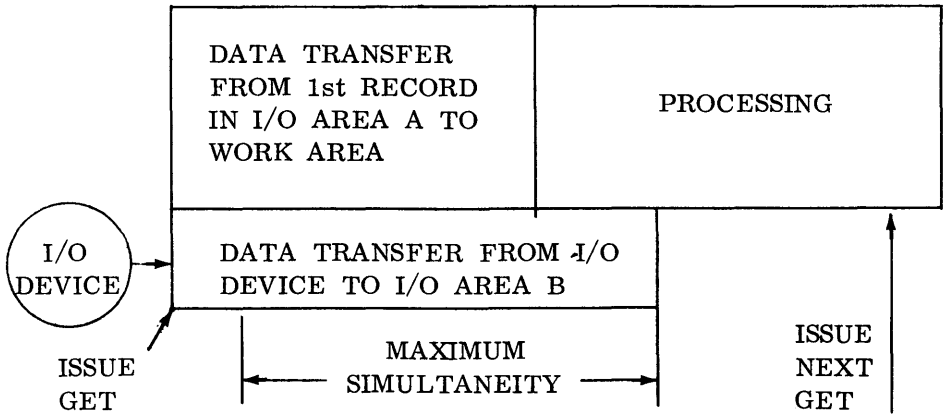
This location is initialized to zero by FCP. If an alternate track is used, the CCHH address of that alternate track is inserted into FILENAME + 48 (through Filename + 51) and is available to the user at I/O termination.

The user can examine this area after I/O termination to determine if an alternate track has been dynamically assigned. Neither DALTA nor Error Recovery resets this area to zero.

Simultaneity *(Cont'd)*

The GET requires time only to point to the record of a block for processing. It is, therefore, possible to process a full block while reading-in the next block.

8. *Blocked Records, Two I/O Areas, One Work Area*



There is no advantage in having a work area when using two I/O areas, because additional simultaneity cannot be achieved, and additional memory is needed for the work area.

Table 1-2. Summary of Achievable Simultaneity

Record Format (Blocked or Unblocked)	Number of I/O Areas	Separate Work Area	Amount of Effective Overlap
UNBLOCKED	1	No	*No overlap.
		Yes	Overlap processing of each record (record move performed by FCP).
	2	No	Overlap processing of each record (no record move required).
		Yes	Overlap processing of each record (no advantage to work area).
BLOCKED	1	No	No overlap.
		Yes	Overlap processing of last record in each block.
	2	No	Overlap processing of full block.
		Yes	Overlap processing of full block (no advantage to work area).

*Overlap of device operation only for buffered devices. No overlap of magnetic tape devices (nonbuffered).

PROGRAMMING I/O USING LOGICAL LEVEL FCP

GENERAL

♦ The logical level FCP provides two I/O accessing methods, serial access (DTFSR) and direct access (DTFDA). The logical level FCP macro instructions and a brief description of their usage are as follows:

File Descriptor Macros

♦ DTFSR - Describes the characteristics of the logical file, indicates the type of processing to be used for the file, and specifies memory areas for the file. This macro is used for those files that require serial access only.

DTFDA - Describes the characteristics of the logical file, indicates the type of processing to be used for the file, and specifies memory areas for the file. This macro is designed primarily for those files that require direct access (records organized in a random order).

DTFEN - Must follow all DTF macros and indicates to the assembler that all files have been described.

I/O Control Macros

♦ I/O Control Macros are divided into three major categories: macros that may be used for both serial access and direct access, macros that may be used for serial access only, and macros that may be used for direct access only.

Serial And Direct Access I/O Control Macros

♦ OPEN - Activates a file in the user's program.

LBRET - Provides the ability to check or write additional standard labels or nonstandard labels.

CLOSE - Deactivates a file that was previously opened.

Serial I/O Control Macros

♦ GET - Obtains the next logical record from a file.

PUT - Sends the next logical record to a file.

RELSE - Allows skipping of remaining records in a block.

TRUNC - Allows writing a short block of records to tape.

CNTRL - Provides physical nondata operations for I/O devices.

PRTOV - Specifies the operation on a printer overflow condition.

FEOV - Forces an end-of-volume condition.

Direct Access I/O Control Macros

- READ - Obtains a physical record from a direct-access device.

  WRITE - Sends a physical record to a direct-access device.

  WAITF - Insures that a previous Read or Write operation has been completed.

  CNTRL (SEEK) - Permits access movement to be overlapped with processing.

MACRO FORMAT

- The format for each of the macro instructions is as follows:

Column

1 7 10 14 16 71 72 73-80

Name		Operation		Operand		
TAG		MACRO		OPERAND1,OPERAND2, OPERANDn		

Some macros have more entries in the Operand field than others. The DTFSR macro has many optional operands, but it is not necessary for a comma to appear for every possible entry. Commas should appear only for those operands that are used. For any of the other macros, it is necessary for a comma to appear for every entry in the Operand field even though an entry is optional and not used.

When the entries in the Operand field require more than one card, any nonblank must be punched in column 72 of all cards of the set for that macro except the last one. A blank column in the Operand field indicates the end of the string of entries for a macro. Columns 73 through 80 may contain an ID sequence number. Continuation cards must begin in column 16.

DTFSR - Define the File in a Serial Type Device

GENERAL DESCRIPTION

- The DTFSR macro describes the logical file, indicates the type of processing to be used for the file, and specifies memory areas for the file. The DTFSR macro is written immediately after the START macro and ahead of the program's source coding. The first card of the macro is called a *header* card and the continuation cards are called *detail entry* cards.

Note:

The DTFSR macro is not written immediately after the START macro if the program is CUP and the SNAPSHOT option has been specified. See TDOS Multichannel Communication System Reference Manual (70-00-612).

FORMAT

◆ The format for DTFSR is given in table 1-3.

Table 1-3. Format for DTFSR Macro

Name (1-7 Characters)	Operation	Operand
Filename	DTFSR	ALTAR = YES, ALTDEV = TAPE, ALTTAPE = nnnnnn, BLKSIZE = n, CKPTREC = YES CONTROL = YES, CRDERR = RETRY, CTLCHR = YES, DEVADDR = nnnnn, DEVICE = { BILLFD | PRINTER | PTAPERD | PUNCH {4 | 6} | READER | TAPE | DISC64 | DRUM65 | MASS68 } , DLYPRX = name, EOFADDR = name, ERROPT = { IGNORE | SKIP | Name } , FILABL = { STD | NSTD | NO } , IOAREA1 = name, IOAREA2 = name, IOREG = n, ISRKEY = n, LABADDR = name, LABNAME = name, MONITOR = YES MRKCTR = nnnn, OVERFLO = YES, OVRTN = name, PRINTOV = YES, READ = { FORWARD | BACK } , RECFORM = { FIXUNB | FIXBLK | VARUNB | VARBLK | UNDEF } ,

FORMAT *(Cont'd)*

Table 1-3. Format of DTFSR Macro *(Cont'd)*

Name	Operation	Operand
Filename (Cont'd)	DTFSR (Cont'd)	RECSIZE=n, REWIND= { UNLOAD / NORWD } , SEEKADR=name, TPMARK=NO, TRANS=name, TRUNCS=YES TYPEFLE= { INPUT / OUTPUT } , UPDATE=YES, USEIN=name, VARBLD=n, VERIFY=YES, WLRERR=name, WORKA=YES,

Note: A comma must *not* follow the last entry.

SPECIFICATION RULES

Name Field

◆ This is a required entry and contains the filename (1 to 7 characters).

Operation Field

◆ DTFSR.

Operand Field

◆ Each operand entry shown in table 1-3 is described below.

ALTAR=YES

◆ This is a required entry for dynamic alternate track assignment for random access. The programmer must not use this entry if this DTFSR has OVERFLO=YES or if update processing (UPDATE=YES) is used for a file that contains records which were written with the physical overflow feature employed.

Note: See Dynamic Alternate Track Assignment.

ALTDEV=TAPE

◆ This optional entry permits a limited amount of device interchangeability. The following combinations are allowed:

DEVICE=PRINTER	PUNCH4/6	READER
ALTDEV=TAPE	TAPE	TAPE

The following restrictions are imposed:

1. Record format must be fixed unblocked.
2. If DEVICE=PRINTER neither a PRTOV nor a CNTRL macro may be issued for the file.
3. The user can not close and then reopen any interchangeable file.

ALTTAPE=nnnnnn

◆ This entry specifies the symbolic name for a magnetic tape device that is used as an alternate when a tape file has two or more reels of data. *This is an optional entry.* The symbolic name for the first magnetic tape device used for the file is specified in the DTFSR entry DEVADDR=nnnnnn.

BLKSIZE=n

◆ The number n indicates the size of the input area or output area specified by IOAREA1. *This is a required entry.* The maximum number of characters transferred to or from the area at any one time must be specified. If two input or output areas are used for a file, they must be of equal size and the size is specified in this entry. If printer output records include control characters and/or variable-length records are specified, this entry must include the area needed for the control character field and/or the block and record size fields. For direct-access files, this size must be the size of the largest data block in the file and must not include the count and key blocks. FCP uses the specified letter n to:

1. Construct the count field of the Channel Command Word (CCW) for an input file.
2. Construct the count field of the Channel Command Word (CCW) for an output file of fixed-length records.
3. Check physical record length for a file of fixed-length blocked input records.
4. Determine if the space remaining in the output area is large enough to accommodate the next variable-length output record (if WORKA = YES is specified).

CKPTREC=YES

◆ This entry is required *if* a tape input file contains checkpoint records interspersed among the data records. With this entry, FCP recognizes the checkpoint records and bypasses them. This entry is ignored for direct access files.

CONTROL=YES

◆ This entry is required *if* a CNTRL macro is to be issued for this file. This entry causes FCP to recognize a CNTRL macro and to generate information for control instructions. A control instruction performs non-data operations such as paper advancing, tape rewinding, etc. When the CONTROL entry is used, the DTFSR entry CTLCHR must not be used.

CRDERR=RETRY

◆ This entry applies only to a card output file and must be specified if error recovery is to be used. When this entry is specified, FCP generates a retry routine and a save area for the card punch record. When an error condition occurs on the 70/234 Card Punch, FCP notifies the operator and then relinquishes control to him. The operator can either terminate the job or instruct FCP to repunch the card. For the 70/236 Card Punch there is no operator intervention or messages. The card is automatically repunched.

CTLCHR=YES

◆ This entry applies only to printer output files. It is required *if* each logical record to be written contains a control character (carriage control). For fixed-length records the control character occupies the first position in the record. For variable-length records, the control character is located after the block and record length control fields. In either case, printing begins with the first character following the control character. When this entry is not specified, any control functions desired must be performed by the CNTRL macro. When the CTLCHR entry is used, the DTFSR entry CONTROL must not be used.

CTLCHR = YES (Cont'd)

Following is a list of printer control bytes, expressed hexadecimally.

Control Byte	Meaning
4n	Space n lines before printing.
0n	Space n lines after printing.
Cn	Skip to channel n (on carriage control tape) before printing.
8n	Skip to channel n (on carriage control tape) after printing.

Note: The range of n is 1 to F except for Skip operations where n must not be 9 or C.

DEVADDR = nnnnnn

◆ This entry specifies the symbolic device name to be associated with this logical file. The symbolic device name must be unique for each logical file and is used by the Executive Control System to assign the actual I/O device to this file. Whenever two devices are used for one logical file (specified in DTFSR ALTTAPE), this entry specifies the symbolic device name for the first device.

DEVICE =

◆ This entry specifies the I/O device type associated with this logical file. *This entry is required.* One of the following entries must be entered:

BILLFD - for an output file printed on the 70/248 Bill Feed Printer.

PRINTER - for an output file printed on the 70/242 or 70/243 Printer.

PTAPERD - for an input or an output file read or punched on the 70/221 Paper Tape Reader/Punch.

PUNCH4 - for an output card file punched on the 70/234 Card Punch.

PUNCH6 - for an output card file punched on the 70/236 Card Punch.

READER - for an input card file read on the 70/237 Card Reader.

TAPE - for an input or output file read from or written to the 70/432, 70/442, or 70/445 Magnetic Tape Devices.

DLYPRX=name (Cont'd)

Since USEIN is required by FCP when the delayed processing option is requested, the programmer can compare the card address from the Communication Table 1-5A with the address of missing cards obtained by the method described above. If the input/output order to be executed cannot address that portion of his file that is unaccessible he should return to FCP to execute the order. Otherwise, the programmer can elect to perform some other function.

When the programmer has requested delayed processing and he gains control at the delayed processing exit address, he can elect to issue the input/output order, or elect not to issue it because he knows through comparison of his address list with the USEIN communication table that the I/O would cause inoperability. To issue the I/O request, the programmer must return to FCP by executing a B 4(0,14). instruction. To bypass the I/O order, the return instruction is BR 14.

EOFADDR=Name

◆ This entry must be specified for card reader files, paper tape reader files, magnetic tape input files, and direct-access files, both input and output. Name specifies the symbolic name of the program end-of-file routine. FCP automatically branches to this routine on an end-of-file condition. For direct-access output files FCP branches to this address if the extents allocated for this file have been exhausted.

FCP detects end-of-file conditions as follows:

1. *Card Reader-* By recognizing /* punched in columns 1 and 2. If cards are allowed to run out without a /* trailer card, an error condition is indicated to the operator.

2. *Paper Tape Reader-* By recognizing the end-of-file characters /* in the first two positions of a data block.

3. *Magnetic Tape Input:*

   Standard Labeled files - by reading a tape mark followed by an EOF label. FCP checks the block count. If LABADDR is not specified, FCP then goes directly to the users EOFADDR. If LABADDR is specified, FCP does not branch to EOFADDR until after allowing the user to process his user labels in his LABADDR routine.

   Nonstandard Labeled Files - two conditions exist depending upon whether or not a LABADDR routine was specified for this file. If the DTFSR entry LABADDR was specified, control is given to the EOFADDR routine only on an end-of-file condition as determined by the LABADDR routine (see LABADDR). If a LABADDR routine is not specified, control is given to the EOFADDR routine when FCP encounters a tape mark. If an end-of-volume condition the user issues an FEOV macro to cause FCP to perform a tape swap. If an end-of-file condition the user issues a CLOSE macro.

   Unlabeled files - by reading a tape mark following the data.

EOFADDR=Name (Cont'd)

4. *Direct-Access Files* - By reading an end-of-file record (a record with data length=0) or by reaching the end of the last extent specified for the file.

DEVICE = (Cont'd)

DISC64 - for an input and/or output file read from or written to the 70/564 Disc Storage Unit.

DRUM65 - for an input and/or output file read from or written to the 70/565 Drum Memory Unit.

MASS68 - for an input and/or output file read from or written to the 70/568 Mass Storage Unit.

DLYPRX=name

◆ *User Facility for Delayed Processing.*

The delayed processing facility is provided for the 70/568 Mass Storage Unit. It provides the programmer with the ability to bypass the processing of portions of his file when that portion is not accessible because of device inoperability, missing cards, etc. He can perform a delayed processing scheme at specific control points.

Note:

This function requires the entry USEIN=name in the DTFDA or DTFSR and the Communication Table generated by the USEIN entry.

To generate the delayed processing facility the following entries must be made in the DTFDA or DTFSR:

DEVICE=MASS 68

DLYPRX=name1

USEIN=name2

When FCP discovers that a 70/568 operation has terminated due to an inoperable condition and the programmer has requested the delayed processing function FCP does the following:

1. Uses the communication table created by the USEIN entry.

2. If the file is output, sets the normal termination indicator, or if it is input, presets the file region to indicate block empty and sets the normal termination indicator. The switches to indicate normal termination and block empty are internal indicators for FCP use.

3. Restores the programmer's registers.

4. Branches to the programmer's delayed processing routine name.

The user program should ask the operator if inoperability was caused either by a missing card or a card that did not select. If so, the user program should store the seek address of the card (bytes 6-12 of Communication Table 1-5A) noting that the card is not selectable for the remainder of the run. The operator can then resume the user program after restoring the device to operable status.

ERROPT = (Cont'd)

Notes:

1. The ERROPT entry does not apply to magnetic tape output files. The job is terminated if a parity error still exists after FCP attempts a standard number of times to write an output block.

2. The user must maintain the integrity of registers 0, 1, 14, 15 while in his ERROPT routine.

3. For random access files, control is returned to this routine when the record cannot be written. If OVERFLOW = YES has not been specified and an attempt has been made to write a record that is larger than the number of bytes remaining this track, then control is returned to this routine. If this routine is not specified the program is terminated.

4. The ERROPT entry does not apply to paper tape.

5. This entry applies to wrong length records if the DTFSR entry WLRERR is not used.

6. If the user issues any FCP macros in his ERROPT routine, the contents of registers 0, 1, 14, 15 must be stored before issuing the macro and then restored.

ERROPT =

◆ This entry applies to direct-access input and output files or magnetic tape input files and specifies functions to be performed for an error block. If a parity error is detected when a block of records is read, the block is reread a standard number of times before the clock is considered an error block. If the reread is unsuccessful, control is given to the operator. The operator can then specify that the reread be attempted again or that control be returned to the program. When control is returned to the program, the ERROPT entry specifies other procedures to be followed on the error condition. If the ERROPT entry has not been specified, the program is terminated. If the ERROPT entry has been specified, it is the address of the user's routine that handles this type of error.

One of the following must be specified:

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. Processing continues with the first record of the next block. However, the error block is included in the block count. This entry is not applicable to direct-access files.

Name - Specifies the address of the program subroutine that may perform any functions required to process the error block. General Register 1 contains the address of the block in error and General Register 14 contains the return address. The program must not issue any GET macros in the routine.

The programmer has the following two options in his ERROPT routine. He can:

1. cause FCP to skip the record containing the parity error and to make the next record available for processing, or

2. cause FCP to ignore the error and process this record.

The two ways to return from the ERROPT = name routine are:

1. to skip this block, the user returns to FCP with a B4 (14) instruction.

2. to ignore the parity error and process this block, the user returns to FCP with a BR 14 instruction.

FILABL =

◆ This entry must be specified for a tape input or output file that contains standard or nonstandard labels. One of the following entries must be specified:

STD - for a tape input file if standard labels are to be checked by FCP, or for a tape output file if standard labels are to be written by FCP. This entry, FILABL = STD, is required for random-access files.

NSTD - for a tape input or output file that has nonstandard labels. These labels may be processed by a program subroutine. NSTD is also used for standard labels that are not to be checked by FCP.

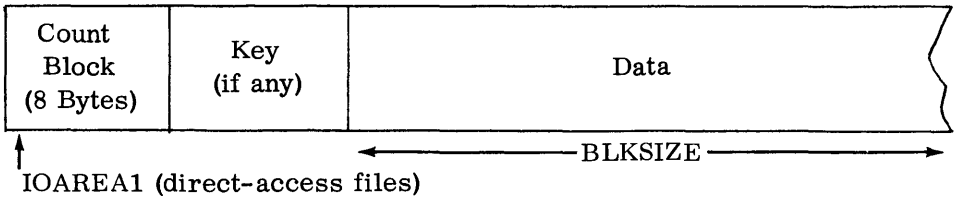
NO - for a tape that does not contain labels. The entry FILABL = NO may be omitted and FCP assumes that there are no labels.

IOAREA1 = Name

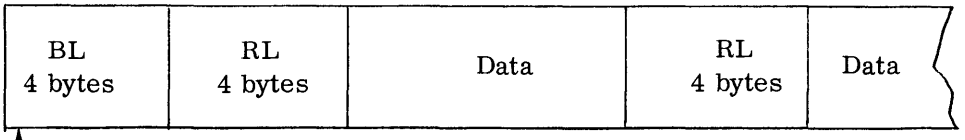
◆ Name specifies the symbolic name of the input or output area used by the file. *This is a required entry.* The I/O routines transfer the records to or from this area. The specified name must be the same as the name used in the DS instruction that the program must set up for this area. For variable records, the I/O area must begin on a halfword boundary and the size of the area must include block and record count fields. For a direct access file, eight bytes must be reserved at the beginning of the I/O area ahead of the positions allotted for the field of the record data. These eight bytes are necessary to allow FCP to process the count field of a record. In addition, the number of bytes required for the key field (if present), must precede the area allotted for the data field of the record. In any case the name of IOAREA1 is the location of the leftmost byte of the count field. Name must not be the name of a DSECT. FCP locates the data portion of a record by referencing IOAREA1 + 8 (allowing eight bytes for the count field) if a key field is not present. If a key field is present (see the DTFSR entry ISRKEY) then FCP locates the data field by referencing IOAREA + k, where k = 8 + key length.

Note: When standard labels are defined for the file, IOAREA1 must be at least 80 bytes in length since FCP reads and writes these labels to and from this area.

For direct-access files, the minimum size that may be specified is 92 bytes if the file contains UHL or UTL labels.



IOAREA1 (direct-access files)



IOAREA1 (Variable-length blocked records on magnetic tape.)

IOAREA2=Name

◆ Two input or output areas can be specified for a file to permit an overlap of data transfer and processing operation. When this is done, the IOAREA2 entry must be included. Name specifies the symbolic name of the second I/O area. Name must not be the name of a DSECT. The rules regarding *variable length records* and *area size* are similar to those specified under IOAREA1 except for tape files. (The label requirement of 80 bytes is applicable only if the DTFSR entry ALTTAPE is specified.)

IOREG=n

◆ This entry specifies the general register that the input/output routines can use to indicate which individual record is available for processing. The letter n indicates one of the General Registers 2 through 13. The other registers cannot be used. FCP puts the absolute base address of the current record in this register each time a GET or PUT is issued.

This entry must be included whenever:

1. Blocked input or output records (from tape to direct-access devices) are processed directly in IOAREA1.

2. Two input or output areas are used and the records (either blocked or unblocked) are processed in the I/O areas.

3. Variable unblocked records are read reverse.

Whenever this entry is included for a file, the DTFSR entry WORKA must be omitted and the GET or PUT macros must not specify work areas. This address reflects the start of the current record which in the case of variable length is the record length field of the record.

ISRKEY=n

◆ This entry must be included for direct-access files if keys are to be read or written. The letter n specifies the number of bytes (1-256) in each key. All keys must be the same length. This entry is used by FCP to determine the location of the data field in IOAREA1. If this entry is omitted, a key length of zero is assumed.

LABADDR=Name

◆ This entry is used for input or output files in conjunction with nonstandard labels when FILABL=NSTD, or with user labels (UHL's and UTL's) when FILABL=STD. Name is the symbolic name of the user program's subroutine that checks or builds the user labels or the nonstandard labels. At the end of this subroutine the user must return to FCP by use of the LBRET macro.

LABADDR = Name (Cont'd)

Input Files - User Header and Trailer Labels (Standard Labels)

1. FCP branches to this user routine on one of the following two conditions:

   a. After processing the standard volume and standard header label sets, or

   b. After processing the end of file or end of volume label set.

2. When the user gets control at his LABADDR address FCP has loaded General Register 1 with the memory location of the user header or trailer label.

3. The user program must determine whether the user label whose address is in General Register 1 is a header or a trailer label.

4. The user program returns to FCP after processing each user header or trailer label, by using the LBRET 2 macro. After each such re-entry into FCP, FCP reads the next user label, puts its address in General Register 1 and goes to the user's LABADDR routine.

5. When the user program returns to FCP via the LBRET 1 macro, or when eight user headers or trailers have been checked, FCP does one of the following:

   a. After header label processing:

      FCP positions the file past the tape mark (tape only) or at the second track of the first extent (direct access).

   b. After trailer label processing:

      If end-of-volume FCP performs end-of-volume processing. If end-of-file FCP branches to the user's EOFADDR routine.

6. If LABADDR has not been specified but user headers and trailers are present, FCP ignores the user labels. If LABADDR is specified, at least one user header and one user trailer must be present.

Output Files - Additional Standard Header Labels (Tape Only)

1. FCP branches to this routine after it has checked the volume label, checked the old standard header label, and has written the new standard header label.

LABADDR=Name (Cont'd)

2. When FCP passes control to the user's LABADDR routine, FCP has loaded General Registers 0 and 1 with the following:

General Register 0 =

Not used			O
Byte	Byte	Byte	Byte

Letter O indicates an additional standard header label is to be written.

General Register 1 = LHE address of where the additional standard header label is to be built.

3. The program is reponsible for building the entire label in the area specified by General Register 1 each time control is received.

4. The program returns to FCP after writing each additional header label by using the LBRET macro. The operand of the LBRET macro indicates to FCP whether the user desires to write another standard label or does not desire to write any more labels.

5. Upon completion of routine processing (all additional header labels have been written), FCP writes a tape mark (tape) or an EOF record (direct-access) following the last additional header label.

Output Files – Additional Standard Trailer Labels (Tape Only)

1. FCP branches to this routine after it has written the standard trailer labels on either an end-of-volume or end-of-file condition.

2. An end-of-volume condition occurs when FCP encounters physical end-of-volume or the program issues a FEOV macro.

3. An end-of-file condition occurs when the program issues a CLOSE macro for the file.

4. When FCP passes control to the user's LABADDR routine, FCP has loaded General Registers 0 and 1 with the following:

General Register 0 =

Not Used			V or F
Byte	Byte	Byte	Byte

V = End-of-volume

F = End-of-file

General Register 1 = LHE address of where the additional trailer label is to be built.

5. The LABADDR routine is responsible for building the entire label in the area specified by General Register 1 each time he receives control.

6. The routine returns to FCP after writing each additional trailer label by using the LBRET macro.

LABADDR=Name (Cont'd)

Input Files – Nonstandard Header Labels (Tape Only)

1. FCP branches to this routine during the OPENing of the file.
2. The program must determine that a header label requires processing as opposed to a trailer label and is responsible for the following:
   a. reading all labels into his own designated areas, by using physical level I/O commands, and
   b. checking his labels, and
   c. positioning the file past the tape mark, if present.
3. The program returns to FCP after processing all nonstandard header labels by using the LBRET macro. The user does not return to FCP with LBRET after each label, as in building or checking additional standard labels, but after all labels have been processed.
4. If the DTFSR entry LABADDR is not specified, FCP assumes a tape mark is present and positions the file past the tape mark.

Input Files – Nonstandard Trailer Labels (Tape Only)

1. FCP branches to the routine after it has sensed the tape mark following the last data block.
2. The program must determine that a trailer label(s) requires processing as opposed to a header label. On trailer label processing, the program must inform FCP of an end-of-volume or an end-of-file condition. This is done by loading General Register 0 with the following information before returning to FCP:

General Register 0 =

Not Used		E	V or F
Byte	Byte	Byte	Byte

EV = End-of-volume

EF = End-of-file

3. The program returns to FCP after processing all nonstandard trailer labels by using the LBRET macro.
4. If the DTFSR entry LABADDR is not specified, FCP passes control to the DTFSR entry EOFADDR which is then responsible for determining end-of-volume or end-of-file (see EOFADDR).

Output Files – Nonstandard Header Labels (Tape Only)

1. The DTFSR entry LABADDR entry must be included for output files that are to contain nonstandard labels. FCP branches to this routine during the OPENing of the file.

LABADDR=Name (Cont'd)

2. When FCP passes control to the user's LABADDR routine, FCP has loaded General Register 0 with the following:

General Register 0 =

Not Used			O
Byte	Byte	Byte	Byte

Letter O indicates nonstandard header label processing is to be performed.

3. The user writes his nonstandard labels by using physical I/O macros. The program is responsible for all label processing in the routine.

4. The program returns to FCP after processing all nonstandard header labels by using the LBRET macro. The user does not return to FCP with LBRET after each label, as in building or checking additional standard labels, but after all labels have been processed.

5. Upon completion of routine processing, FCP writes a tape mark following the last nonstandard header label unless the DTFSR entry TPMARK=NO has been specified for the file.

Output Files - Nonstandard Trailer Labels (Tape Only)

1. FCP branches to this routine on either an end-of-file or an end-of-volume condition.

2. An end-of-volume condition occurs when FCP encounters physical end-of-volume or the program issues a FEOV macro.

3. An end-of-file condition occurs when the program issues a CLOSE macro for the file.

4. When FCP passes control to the user's LABADDR routine, FCP has loaded General Register 0 with the following:

General Register 0 =

Not Used			V or F
Byte	Byte	Byte	Byte

V = End-of-volume

F = End-of-file

5. On both end-of-volume and end-of-file processing the user is responsible for creating and writing all nonstandard labels. Tape marks are written by FCP before and after the nonstandard trailer labels.

6. The program returns to FCP after processing all labels by using the LBRET macro.

LABADDR=Name (Cont'd)

General Considerations - When a LABADDR routine is specified for a file (input or output), the routine is responsible for both header and trailer label processing associated with that file. The routine receives control from FCP at the entrance location for both header and trailer label processing. The routine, in turn, must determine which type of label requires processing and take appropriate action accordingly.

LABNAME=Name

◆ For tape files, this entry indicates the location of the file's volume and standard labels. For direct-access files, this entry indicates the location of the area where FCP builds the file's extent matrix. For *tape files* and *direct-access files* this is an *optional* entry.

1. *Tape Files* - this entry applies only to files containing standard labels. It indicates to FCP the area where the file's standard Header label is located. From the information contained in this area, FCP checks the file's volume label and checks or builds the file's standard header and trailer labels. When this entry is used, the information must be available in the area prior to the OPENing of the file. The user may modify this area at object time before OPENing the file.

   The format for the standard label information area is as follows:

Bytes	Contents
0-6	Filename-one to seven character alphanumeric field which contains the name of the file.
7-75	For input files, fields 3 to 10 of the file header label. For output files, fields 3 to 13 of the file header information. Unused positions should be set to $(40)_{16}$.
76-81	Reserved for use by FCP.

If this entry is omitted the program must enter label information during program initiation by using run-time label parameter entries (VOL and TPLAB). During the OPENing of the file, FCP locates the label information for this file and inserts the applicable address into the LABNAME location of the DTFSR expansion.

Note: When the LABNAME entry is included for a tape file, run-time label parameter entries cannot be entered for the file.

LABNAME=Name (Cont'd)

2. *Direct-Access Files* - LABNAME points to the area where FCP builds the file's extent matrix. The area must provide sufficient memory space to hold the address matrices for as many extents as required by the file.

   The user has the ability, however, to describe or override the size of the matrix area at object time.

   If the LABNAME entry is omitted, a run-time parameter VDC statement must be entered defining the size of the matrix.

   If the LABNAME entry is used, the user still has the ability at object time to enter run-time parameter VDC statements to override the LABNAME matrix size.

   The format of the extent matrix is indicated below. This format is repeated for each volume.

Bytes	Contents
0-5	Volume Serial Number
6-7	Device Assignment Halfword
8-9	Bin Number (Binary)
10	Volume Sequence Number (Binary)
11	Number of Bytes in this Entry
12	Last Extent Indicator
13-25	Extent Description No. 1
13	Extent Number
14-17	LHE of Extent (CCHH)
18-21	RHE of Extent (CCHH)
22-25	Last Relative Track Number (HHHH)

Notes: The last 13 bytes must be repeated for each additional extent (up to 16) per volume.

The size of the extent matrix depends upon two factors; the number of volumes that this file crosses and the number of extents (file areas) on each volume.

MONITOR=YES

◆ This is an *optional* parameter. This entry is specified only if DEVADDR=SYSIPT, SYSOPT, or SYSLST and only if the user wants the optional ability to run his program under Monitor.

When this parameter is present, the following DTFSR entries are generated, regardless of how the user has defined them:

If DEVADDR=SYSIPT or SYSOPT

DEVICE=READER or PUNCH

IOAREA2 (omitted)

WORKA (omitted)

IOREG (omitted)

RECFORM=FIXUNB

If DEVADDR=SYSLST

DEVICE=PRINTER

CONTROL (omitted)

CTLCHR=YES

PRTOV (omitted)

IOAREA2 (omitted)

RECSIZE=N (register number to contain size of each output record).

RECFORM=UNDEF

WORKA (omitted)

IOREG (omitted)

If DEVADDR is other than SYSIPT, SYSOPT, or SYSLST, the entry MONITOR=YES is not allowed.

If this entry is omitted and DEVADDR=SYSIPT, SYSOPT, or SYSLST then the files are defined as the user has specified, but the program cannot be run under MONITOR.

When this entry is included and DEVADDR=SYSIPT, SYSOPT, or SYSLST the program can be run either under Monitor control or not.

The program being run under Monitor can issue either GET's and PUT's or Monitor macros to the devices SYSIPT, SYSOPT, SYSLST.

MRKCTR-nnnn

◆ This is an *optional* tape entry and, if *omitted*, FCP assumes the file is positioned properly when the file in OPENed. If used, nnnn is a one to four character numeric field which indicates the number of tape marks the tape is to be forward spaced. The positioning takes place during the OPENing of the file.

READ=

◆ This entry can be included for a tape input file to specify the direction in which the tape is to be read. One of the following can be specified:

FORWARD - For a magnetic tape read in the forward direction.

BACK - For a magnetic tape read in the backward direction. A tape file may be read backwards only if it contains unblocked records, fixed-length blocked records, or undefined records. Variable-length blocked records are not permitted to be read in a backward direction. Also, multivolume files are not permitted to be read backwards.

Note: If this entry is omitted, FCP assumes forward reading.

OVERFLO=YES

◆ This entry applies only to direct-access files and is required if it is desired to use the record overflow feature.

The record overflow feature is an equipment feature that allows for a record to be written on more than one track. Using this feature, up to a full cylinder can be written with one PUT.

When this entry is included, FCP performs the following depending on the file type:

1. accesses and debatches overflow records.
2. updates overflow records.
3. creates overflow records.

When FCP creates an overflow record, it ensures that it fits in the cylinder, that no segments fall on defective tracks, and that no segments fall on alternate tracks. In addition, during the creation of overflow records, FCP ensures that no segments are written on a track that had already been partially filled. If any of these conditions occur, the overflow record is not written and control is transferred to the ERROPT=name address. If this entry is not specified, the program is terminated.

OVRTN=Name

◆ This is an *optional* end-of-volume routine for magnetic tape and serial direct access files. Name specifies the address of a user program routine to which FCP transfers control following completion of its end-of-volume logic. When the user routine has received control, FCP has completed the processing of the header label(s) for the file's next volume. Within the OVRTN routine the user must not issue any FCP macros. The user must return control to FCP by transferring to the address that FCP has stored in General Register 14.

If a CKPT is issued within the routine, the restart address location, as specified in the first operand of the CKPT macro, must be within the OVRTN routine. That is, at restart, the program must return to the OVRTN routine at a point after the CKPT in order that the program can return control to FCP by using General Register 14.

Notes:

1. This routine is for serial processing only.
2. Control is not transferred to this routine when an FEOV is issued.
3. This routine is not applicable to DTFPH files.

PRINTOV=YES

◆ This entry must appear whenever the PRTOV (Printer Overflow) macro is included in the program. When this entry is specified, one more line is printed after sense of channel 9 or 12 before skipping to channel 1.

RECFORM=

◆ This *required* entry specifies the type of records in the input or output file. One of the following entries must be specified:

FIXUNB - For fixed-length unblocked records, applicable to any file.

FIXBLK - For fixed-length blocked records, applicable to magnetic tape or direct-access files only.

VARUNB - For variable-length unblocked records, applicable to magnetic tape and direct-access input or output files; card punch output and printer output files *only*.

VARBLK - For variable-length blocked records, applicable to magnetic tape and direct-access files only.

UNDEF - For undefined records, applicable to any file *except* card input.

Table 1-4. Device-Record Types Table

Device	FIXUNB	FIXBLK	VARUNB	VARBLK	UNDEF
Magnetic Tape	x	x	x	x	x
Direct Access	x	x	x	x	x
Card Input	x				
Card Output	x		x		x
Printer Output	x		x		x
Paper Tape Input	x				x
Paper Tape Output	x				x

x = Valid specification

RECSIZE=n

◆ This entry is required for files defined as containing fixed-length blocked or undefined records (RECFORM=FIXBLK or UNDEF).

Fixed-length Block Records - The letter n specifies the number of characters in an individual record. The I/O routines use this factor for blocking and deblocking records and for checking record length of input records. The minimum number of characters that can be specified for an individual record is 12; the maximum is 65,536.

Undefined Records - The letter n specifies the number of the General Registers 2-13 may be used. When the user issues a GET, FCP reads the record and loads the record size into register n before returning control to the user. On output files the user must load register n with the record size before issuing the PUT.

REWIND=

♦ If this optional entry is not used, tape files are automatically rewound, but not unloaded on an OPEN or CLOSE macro and an end-of-volume condition. If other operations are desired, one of the following operands must be specified:

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.

SEEKADR=Name

♦ This entry must be included for direct-access files if CONTROL=YES is specified. It indicates the location of the track reference field.

The track reference field is a seven-byte field. The symbolic name refers to the leftmost byte. The address in the field is an actual track address.

The format of the actual track reference field is shown in table 1-5.

Table 1-5. Actual Track Reference Field

Byte Number	Identification	Meaning
0	M	This byte contains the binary volume sequence number on which the record is located. All volumes for a file are numbered consecutively starting with 0.
1-2	BB	Not used.
3	C	If DEVICE=MASS68, this byte contains the actual card number on which the record is located. For the other random access devices, this byte is $(00)_{16}$.
4	C	This byte contains the actual cylinder number in which the record is located.
5	H	This byte is $(00)_{16}$.
6	H	This byte contains the head number (the track within the cylinder) that is used to reference the record.

TPMARK=NO

♦ This optional entry indicates the absence of a tape mark before the first data block on an output file with omitted labels.

For output files containing nonstandard labels TPMARK=NO is included to prevent FCP from writing a tape mark between the header label set and the data records. This entry is not applicable to the terminal tape mark following the data records, which is always required.

This entry has no meaning for input files as it is up to the user to position the file past the nonstandard header labels and the initial tape mark if it is present.

TRANS=Name

♦ This optional entry applies only to a paper tape input file. For paper tape input files recorded in a mode other than EBCDIC, the code must be translated to EBCDIC code for use in the Spectra system.

Six, Seven, Eight (6, 7, 8) Channel Tape - Name specifies the symbolic name of a translation table (the table must be word oriented). FCP uses this table to translate the paper tape code and then makes the record available to the program in usable form directly in the input area or in the work area if specified. Name is the address of the leftmost byte of the table.

Five (5) Channel Tape – This entry does not apply. The user must do his own translation.

TRUNCS=YES

♦ This entry is required for direct-access files containing fixed-length blocked records if short blocks are to be processed. It must be included:

1. For an output file if the TRUNC macro is to be issued in the program.

2. For an input file if the TRUNC macro was issued to write short blocks when the file was originally created.

This entry must be included for tape output files if variable-length records are to be built in the work area.

TYPEFLE=

♦ This entry must be included to specify the type of file (input or output). One of the following entries must be specified:

INPUT - must be specified for magnetic tape or direct access input files, card reader files, paper tape input files.

OUTPUT - must be specified for magnetic tape or direct-access output files, card punch files, printer output files, and paper tape output files.

VARBLD=n

◆ Whenever variable-length blocked records are built directly in the output area (no work area specified), this entry must be included. The letter n specifies the number of a general register which contains the length of the available space remaining in the output area. General Registers 2 through 13 may be specified. After the PUT macro is issued for a variable-length record, FCP calculates the space still available in the output area and supplies it to the program in the register specified. The program then compares the length of the next variable length record with the available space to determine if the record fits in the area. This check must be made before the record is built. If the record does not fit, the program must issue a TRUNC macro to transfer the completed block of records to the tape file. The present record is then built at the beginning of the output area as the first record in the next block.

VERIFY=YES

◆ This entry should be included if records are to be checked after they are written to disc or drum (DEVICE=DISC64 or DRUM65). When this entry is included and a record does not verify, control is returned to the program's error routine (see ERROPT). If no error routine is specified the program is terminated.

Note: This entry does not apply to the 70/568 Mass Storage Unit.

WLRERR=Name

◆ This entry applies only to tape and direct-access input files. Name specifies the symbolic name of a program subroutine to which FCP branches if a wrong-length record is read. In the subroutine, any operation may be performed with the exception of a GET macro. The address of the record is supplied by FCP in General Register 1. At the end of the routine, the program must return to FCP by branching to the address in General Register 14.

Whenever fixed-length blocked records or variable-length records are specified, the machine check for wrong-length records is suppressed, and FCP generates a program check of record length. For fixed-length blocked records, record length is considered incorrect if the physical tape or direct access record (gap-to-gap) that is read is not a multiple of the logical record length (specified in DTFSR entry RECSIZE), up to the maximum length of the block (specified in DTFSR entry BLKSIZE). This permits the reading of short blocks of logical records without a wrong-length record indication. For variable-length records, record length is incorrect if the length of the tape or direct-access record is not the same as the block length specified in the block count control field. For fixed-length unblocked records, FCP checks for a wrong length record indication that may have been set as a result of an I/O operation.

UPDATE = YES

◆ This entry is required if a direct-access file (TYPEFLE = INPUT) is to be updated. That is, direct-access records are to be read, processed, and then transferred back to the same direct-access record locations from which they were read. If this entry is included only one IOAREA may be specified and no work area.

USEIN = name

◆ *User Exit on Initiation*

This facility provides the programmer with the ability to obtain control from FCP whenever FCP initiates a physical write to the file. Thus the programmer can create a backup file which can be used to facilitate reconstruction of a file.

To generate the user Exit routine the programmer must specify in his DTFDA or DTFSR the keyword USEIN = name. This causes FCP to create a 32-byte communication table (Table 1-5A) and transfer control to the user routine immediately prior to executing a physical write order to the random access device. FCP builds this communication table beginning at the user's USEIN address.

The programmer's USEIN subroutine must be a physical level operation and must maintain the integrity of registers 0, 1, 14, and 15. The users returns control back to FCP by use of a BR 14 instruction. FCP then completes the write operation and normal processing continues. No logical FCP macros can be executed within the user routine and the routine must be double word aligned (example: Name DS OD).

Table 1-5A. FCP Communication Table

Byte	Meaning
0 - 5	Volume Serial Number.
6 - 7	Bin Number.
8 - 12	CCHHR Record Indentifier (R Supplied if available).
13 - 15	Key and Data Length if Op Code = 83 (write CKD).
16	CCW Operation Code.
17 - 19	CCW Data Address.
20	CCW Flag Byte.
21	0. Reserved.
22 - 23	CCW Byte Count.
24 - 27	Standard DeviceByte and Sense Byte Information.
28 - 29	00. Reserved.
30 - 31	Device Assignment Halfword.

WLRERR=Name (Cont'd)

Notes:

1. If the DTFSR entry for ERROPT is specified for this file the wrong length record will be treated as an error block and handled according to the users specifications for ERROPT (IGNORE, SKIP, or NAME.)
2. If both the WLRERR and ERROPT entries are omitted, the program will be terminated on a wrong length condition.
3. If any FCP macros are issued in the WLRERR routine, then the contents of registers 0, 1, 14, 15 must be stored before issuing the macro and then be restored thereafter.
4. The WRRERR routine does not apply to undefined records.

WORKA=YES

◆ Input/output records can be processed or built in work areas instead of the input/output areas. If this is done, the WORKA=YES entry must be included, and the program must define the work area in memory. The symbolic name used in the DS instruction to reserve the work area must be specified in each GET or PUT macro. On a GET or PUT macro, FCP moves the record to or from the specified work area. Whenever this entry is used, the DTFSR entry IOREG must be omitted.

Table 1-6. DTFSR Entry Summary Table

Name	Operation	Operand	70/248 Bill Feed Printer	70/242, 243 Printer	70/221 Paper Tape Reader Punch	70/234, 236 Card Punch	70/237 Card Reader	70/432, 442, 445 Magnetic Tape Devices	70/564 Disc Storage Unit	70/565 Drum Memory Unit	70/568 Mass Storage Unit	Remarks
Filename	DTFSR		X	X	X	X	X	X	X	X	X	
		ALTAR = YES							*	*	*	See DTF entry for rules regarding use.
		ALTDEV = TAPE		*		*	*					
		ALTTAPE = nnnnnn						*				
		BLKSIZE = n	X	X	X	X	X	X	X	X	X	
		CKPTREC = YES						*				
		CONTROL = YES	*	*				*	*	*	*	
		CRDERR = RETRY				*						
		CTLCHR = YES	*	*								
		DEVADDR = nnnnnn	X	X	X	X	X	X	X	X	X	
		DEVICE =										
		BILLFD	X									
		PRINTER		X								
		PTAPERD			X							
		PUNCH4				X						
		PUNCH6				X						
		READER					X					
		TAPE						X				
		VIDEOSC										
		DISC64							X			
		DRUM65								X		
		MASS68									X	
		DLYPRX = Name									*	Required for Delayed Processing.
		EOFADDR = Name			X		X	X	X	X	X	Input files only.
		ERROPT =										
		IGNORE						*	*	*	*	R/A input and output files or tape input files only.
		SKIP										
		Name										
		FILABL =										
		STD						X				Standard or Nonstandard. Labels only.
		NSTD										
		NO										
		IOAREA1 = Name	X	X	X	X	X	X	X	X	X	
		IOAREA2 = Name	*	*	*	*	*	*	*	*	*	Required if no work
		IOREG = n	*	*	*	*	*	*	*	*	*	area is specified for
		ISRKEY = n							*	*	*	blocked records.
		LABADDR = Name						*	*	*	*	Additional Standard or Nonstandard Labels.
		LABNAME = Name						*	*	*	*	
		MONITOR = YES		*		*	*					
		MRKCTR = nnnn						*				
		OVERFLO = YES							*	*	*	
		OVRTN = Name						*	*	*	*	
		PRINTOV = YES	*	*								Required if PRTOV macro is used.

X = required entry. * = optional entry.

Table 1-6. DTFSR Entry Summary Table *(Cont'd)*

Name	Operation	Operand	70/248 Bill Feed Printer	70/242, 243 Printer	70/221 Paper Tape Reader Punch	70/234, 236 Card Punch	70/237 Card Reader	70/432, 442, 445 Magnetic Tape Devices	70/564 Disc Storage Unit	70/565 Drum Memory Unit	70/568 Mass Storage Unit	Remarks
Filename (Cont'd)	DTFSR (Cont'd)	READ= FORWARD BACK						*				Input file only.
		RECFORM= FIXUNB	X *	X *	X *	X *	X *	X *	X *	X *	X *	
		FIXBLK						*	*	*	*	
		VARUNB	*	*		*		*	*	*	*	
		VARBLK						*	*	*	*	
		UNDEF	*	*	*	*		*	*	*	*	
		RECSIZE=n	X	X	X	X		X	X	X	X	Fixed-length blocked or Undefined records.
		REWIND= UNLOAD NORWD						*				
		SEEKADR=Name							*	*	*	Required if CONTROL=YES.
		TPMARK=NO						*				Nonstandard or Omitted Labels.
		TRANS=Name			*							Paper Tape Input recorded in mode other than EBCDIC.
		TRUNCS=YES							*	*	*	Required for short block processing.
		TYPEFLE= INPUT OUTPUT	X	X	X	X	X	X	X	X	X	
		UPDATE=YES							*	*	*	Required for update processing.
		USEIN=Name							*	*	*	
		VARBLD=n			*			*	*	*	*	Required if no work area is specified for VARBLK records. Paper Tape Output only.
		VERIFY=YES							*	*	*	
		WLRERR=Name						*	*	*	*	Input file only.
		WORKA=YES	*	*	*	*	*	*	*	*	*	Required if no IOREG is specified for blocked records.

X = required entry. * = optional entry.

DTFDA Define the File for Direct Access

GENERAL DESCRIPTION

◆ The DTFDA macro describes the logical file, indicates the type of processing to be used for the file, and specifies the memory area for the file. The DTFDA macro is written immediately after the START macro and ahead of the program's source coding. The first card of the macro is called a *header* card and the continuation cards are called *detail entry* cards.

Note:

The DTFDA macro is not written immediately after the START macro if the program is CUP and the SNAPSHOT option has been specified. See TDOS Multichannel Communication System Reference Manual (70-00-612).

FORMAT

◆ The format for DTFDA macro is given in table 1-7.

Table 1-7. Format of DTFDA Macro

Name (1-7 Characters)	Operation	Operand
Filename	DTFDA	ALTAR=YES, AFTER=YES, AFTERID=YES, BLKSIZE=n, CAPREC=YES, CONTROL=YES, DEVICE={DISC64 / DRUM65 / MASS68}, DLYPRX=name, ERRBYTE=name, IDLOC=name, IOAREA1=name, KEYARG=name, KEYLEN=n, LABADDR=name, LABNAME=name, READID=YES, READKEY=YES, RECFORM={FIXUNB / UNDEF}, RECSIZE=n, RELADDR=YES, SEEKADR=name, SRCHM=YES, TYPEFLE={INPUT / OUTPUT}, USEIN=name, VERIFY=YES, WRITEID=YES, WRITEKY=YES,

SPECIFICATION RULES

Name Field

◆ This is a required entry and contains the file name.

Operation Field

◆ DTFDA.

Operand Field

◆ Each operand entry shown in table 1-7 is described below.

AFTER = YES

◆ With this entry, FCP provides the ability to add a new record after the last existing record on a specified track. When a WRITE AFTER is issued, the remainder of the specified track is erased. If this entry is included, IOAREA1 must contain an additional eight bytes in which FCP calculates the count field of the new record.

AFTERID = YES

◆ With this entry, FCP provides the ability to add a new record after a record with a specified ID. When a WRITE AFTERID is issued, the remainder of the specified track is erased. If this entry is included, IOAREA1 must contain an additional eight bytes in which FCP calculates the count field of the new record. If DEVICE = MASS68, only ID = 0 can be specified.

BLKSIZE = n

◆ The letter n indicates the size of the input area or output area specified by IOAREA1. *This is a required entry.* The maximum number of characters that are transferred to or from the area at one time must be specified. If the key field is to be transferred to or from memory, the size of the area must be large enough to contain the key and the record. If a file is to be created or if records are to be added to a file, an additional eight bytes must be specified. These eight bytes contain the count field. If DEVICE = MASS68, maximum block size is 2048 bytes, including keysize.

CAPREC = YES

◆ This entry causes the capacity record option to be used by the program. This entry must be specified if the WRITEAFTER macro is used. The capacity record option provides an efficient means of maintaining an inventory of the unused space on each track in the file. The first record on each track (record zero) is used to contain the number of unused bytes on the track and the ID of the last record on the track. This record is called the capacity record. When records are to be added to a track, FCP uses the information in the capacity record to insure that the record to be added fits. If it does, FCP writes the record and automatically updates the capacity record. If the record does not fit, the no-room-found indicator is set in the error/status byte (see ERRBYTE). The format of the capacity record is as follows:

	Count Field					Data Field		
	ID					ID of Last Record	No. of Unused Bytes	Resv.
	CC	HH	R	KL	DL			
No. of Bytes	2	2	1	1	2	5	2	1

DLYPRX=Name (Cont'd)

Note: This function requires the entry USEIN = name in the DTFDA or DTFSR and the Communication Table generated by the USEIN entry.

To generate the delayed processing facility the following entries must be made in the DTFDA or DTFSR:

DEVICE = MASS68

DLYPRX = name1

USEIN = name2

When FCP discovers that a 70/568 operation has terminated due to an inoperable condition and the programmer has requested the delayed processing function FCP does the following:

1. Uses the communication table created by the USEIN entry.
2. If the file is output, sets the normal termination indicator, or if it is input, presets the file region to indicate block empty and sets the normal termination indicator. The switches to indicate normal termination and block empty are internal indicators for FCP use.
3. Restores the programmer's registers.
4. Branches to the programmer's delayed processing routine name.

The user program should ask the operator if inoperability was caused either by a missing card or a card that did not select. If so, the user program should store the seek address of the card (bytes 6-12 of Communication Table 1-5A) noting that the card is not selectable for the remainder of the run. The operator can then resume the user program after restoring the device to operable status.

Since USEIN is required by FCP when the delayed processing option is requested, the programmer can compare the card address from the Communication Table 1-5A with the address of missing cards obtained by the method described above. If the input/output order to be executed cannot address that portion of his file that is unaccessible he should return to FCP to execute the order. Otherwise, the programmer can elect to perform some other function.

When the programmer has requested delayed processing and he gains control at the delayed processing exit address, he can elect to issue the input/output order, or elect not to issue it because he knows through comparison of his address list with the USEIN communication table that the I/O would cause inoperability. To issue the I/O request, the programmer must return to FCP by executing a B 4(0,14). instruction. To bypass the I/O order, the return instruction is BR14.

ERRBYTE=Name

◆*This entry is required.* Name specifies the symbolic name of a two-byte area that FCP uses to supply information concerning the terminating conditions of an I/O operation. The specified name must be the same as the name used in the DS instruction that the program must set up for this area. The information contained in this two-byte area (Error/Status field) is available to the program after a WAITF macro has been executed. The bit significance of the error/status field is shown in tables 1-8 and 1-9. A WAITF must be issued after Read or Write operations to insure that the operation has been completed successfully.

CAPREC = YES (Cont'd)

Legend:

Count Field

ID = identifier composed of CC (Cylinder Number), HH (Head Number) and R (Record Number). The record number of the capacity record is always $(00)_{16}$.

KL = the number of bytes in the key field of the record. Because the capacity record does not contain a key field, KL is always $(00)_{16}$.

DL = the number of bytes in the data field of the record. Because the data field of the capacity record is always eight bytes, DL is always $(0008)_{16}$.

Data Field

Bytes 1-5 - contain the Identifier (CCHHR) of the last record on the track.

Bytes 6-7 - contain a binary count of the unused bytes on the track.

Byte 8 - reserved for future use.

CONTROL = YES

◆ This entry is required if a CNTRL macro is to be issued for this file. This entry causes FCP to recognize a CNTRL macro and to generate information for control instructions. The CNTRL macro is used to perform Seek operations.

DEVICE =

◆ This entry is required and specifies the I/O device type associated with this logical file. One of the following must be entered:

DISC64 - for an input and/or output file on the 70/564 Disc Storage Unit.

DRUM65 - for an input and/or output file on the 70/565 Drum Memory Unit.

MASS68 - for an input and/or output file on the 70/568 Mass Storage Unit.

DLYPRX = Name

◆ *User Facility for Delayed Processing.*

The delayed processing facility is provided for the 70/568 Mass Storage Unit. It provides the programmer with the ability to bypass the processing of portions of his file when that portion is not accessible because of device inoperability, missing cards, etc. He can perform a delayed processing scheme at specific control points.

ERRBYTE=Name (Cont'd)

Table 1-8. Bit Significance of Error/Status Field (Byte 1)

Bit Position	Name	Meaning
0	File Protect Error	This bit is set if a file protection error occurs. When a read or write macro is executed, the cylinder address supplied in the track reference field is checked against the extent matrix. (The extent matrix is built for the entire file as part of the open processing and contains the boundaries of each extent in the file.) If the cylinder address specified is outside of the file limits, the file protect bit is set.
1	Wrong Length Record	This bit is set if a READID, WRITEID, READKEY, or WRITEKEY macro is executed and the input/output block size is greater than the key length and/or data length as specified in the count field of the record. (The CCW byte count has not lapsed.)
2,3	Reserved	These bits are reserved for future use.
4	No Room Found	This bit can be set only if a WRITE AFTER or WRITE AFTERID macro is issued and the CAPREC=YES entry is specified in the DTFDA. This bit is set if the record to be written is too large to fit in the remaining unused portion of the specified track. The record is not written.
5,6,7	Reserved	These bits are reserved for future use.

Table 1-9. Bit Significance of Error/Status Field (Byte 2)

Bit Position	Name	Meaning
0	Data Check in Count Area	This bit is set when the cyclic check bytes in the count field do not verify when the count field of a record is read (or written - 70/568). Error recovery has been attempted but was unsuccessful. The count field of a record is read when the following macros are issued: READKEY READID WRITEKEY WRITEID

ERRBYTE=Name (Cont'd)

Table 1-9. Bit Significance of Error/Status Field (Byte 2) *(Cont'd)*

Bit Position	Name	Meaning
1	Track Overrun	This bit can be set only if a WRITE AFTERID macro is executed and the CAPREC=YES entry is *not* specified in the DTFDA. This bit is set if the record to be written is too large to fit in the remaining unused portion of the specified track. FCP writes the portion of the record that fits and the remainder of the record is lost.
2	End of Cylinder	This bit can be set only if a READKEY or WRITEKEY macro is executed and the SRCHM=YES entry is specified in the DTFDA. This bit is set if end of cylinder is detected and the specified record has not been found.
3	Data Check Reading (or Writing - 70/568) Key or Data	This bit is set when the cyclic check bytes in the key or data fields do not verify. (Error recovery has been attempted but was unsuccessful.)
4	No Record Found	This bit can be set only if a READKEY READID, WRITEKEY, or WRITEID macro is executed and the SRCHM=YES entry is not specified in the DTFDA. This bit is set if end of track is detected and the specified record has not been found.
5	End of File	This bit is set if a count field is read, containing a data length (DL) of zeros.
6,7	Reserved	These bits are reserved for future use.

IDLOC=Name

◆ This optional entry specifies that FCP is to supply the ID (CCHHR) of the record just processed after each READKEY, WRITEKEY, or WRITEAFTER macro is executed. Name specifies the symbolic name of the five-byte area that FCP used to store the ID. The program must reserve this area.

IOAREA1=Name

◆ Name specifies the symbolic name of the input or output area used by this file. *This is a required entry.* Name must not be the name of a DSECT.

The I/O routines transfer the records to or from this area. The specified name must be the same as the name used in the DS instruction that the program must set up for this area. IOAREA1 must be large enough to contain the maximum number of bytes that are required in any READ or WRITE operation. If the key field is to be read or written, the size of the area must be large enough to contain the key and the data field of the record. (The length of the key is specified in KEYLEN=n). If a file is to be created or if records are to be added to a file, IOAREA1 must contain an additional eight bytes. FCP uses these eight bytes to construct the count field of the record.

Whenever a Read or Write macro is executed, FCP assumes that IOAREA1 contains the information implied by the type of I/O macro that is executed (see below).

Macro	IOAREA1 Contents	
	KEYLEN specified	KEYLEN Not specified
READKEY	Data	(invalid)
READID	Key, Data	Data
WRITEKEY	Data	(invalid)
WRITEID	Key, Data	Data
WRITE AFTERID	Count, Key, Data	Count, Data
WRITE AFTER	Count, Key, Data	Count, Data

Note:

1. Only one IOAREA can be specified in the direct-access method. The minimum size that can be specified is 92 bytes if the file contains UHL or UTL labels; IF DEVICE=MASS68, the minimum is 2056 bytes.
2. A work area cannot be specified.

KEYARG=Name

◆ This entry is required if the READKEY or WRITEKEY macros are used in the program. Name specifies the symbolic name of the area in which the program places the key to be used for the READKEY and WRITEKEY operations. The specified name must be the same as the name used in the DS instruction that the program must set up for this area.

KEYLEN=n

◆ The letter n indicates the number of bytes in the key field. This entry is required if records are to be referred to by key or if the key field is to be read or written. All keys in the file must be the same length. The maximum length of the key field is 255 bytes. If KEYLEN is omitted, keys are ignored, whether they exist or not. When record reference is by key, FCP uses this entry to construct the count field of the record. FCP also uses this entry in conjunction with IOAREA1 to determine where the data field in IOAREA1 is located.

LABADDR=Name

♦ This entry is used when the program requires one or more standard labels in addition to the standard file header label. Name indicates the symbolic name of the program's subroutine that checks or builds the additional labels. FCP branches to this subroutine after it has processed the standard header label. At the end of this subroutine, the program must return to FCP by use of the LBRET macro.

FCP processes the labels as follows:

Input Files – Additional Standard Header Labels

1. FCP branches to this routine after it has checked the volume and header labels.

2. When control is received by the routine, General Register 1 contains the memory location of the additional user header label.

3. The program must determine that a header label (as opposed to a trailer label) requires processing.

4. The program returns to FCP after processing each additional header label by using the LBRET macro.

5. If the LABADDR entry is omitted, additional standard labels are ignored.

Output Files – Additional Header Labels

1. FCP branches to the LABADDR routine after it has checked the volume label, checked the old standard header label and has written the new standard header label.

2. When control is received by the routine, General Registers 0 and 1 contain the following information:

General Register 0 =

Not Used			O
Byte	Byte	Byte	Byte

Letter O indicates an additional standard header label is to be written.

General Register 1 = LHE address of where the additional standard header label is to be built.

3. FCP places UHLI in the first four bytes of the area where the first additional standard header label is to be built. This identification is increased by one (UHL2, UHL3, etc.) for each additional label to be written.

LABADDR=Name (Cont'd)

4. The program is responsible for building the entire 76-byte label in the area specified by General Register 1 each time he receives control.

5. The program returns to FCP after processing each label by use of the LBRET macro.

LABNAME=Name

♦ Name specifies the symbolic name of the first byte of the extent matrix area. *This is an optional entry.* This area is used by the OPEN routine to build a matrix containing the limits of each extent in the file. The specified name must be the same as the name used in the DS instruction that the program must set up for this area.

LABNAME points to the area where FCP builds the file's extent matrix. The area must provide sufficient memory space to hold the address matrices for as many extents as required by the file.

The user has the ability, however, to describe or override the size of the matrix area at object time.

If the LABNAME entry is omitted, a run-time parameter VDC statement must be entered defining the size of the matrix.

If the LABNAME entry is used, the user still has the ability at object time to enter run-time parameter VDC statements to override the LABNAME matrix size.

The format of the extent matrix is indicated below:

Bytes	Contents
0-5	Volume Serial Number
6-7	Assignment Halfword
8-9	Bin Number (Binary)
10	Volume Sequence Number (Binary)
11	Number of Bytes in this Entry
12	Last Extent Indicator
13-25	*Extent Description No. 1*
13	Extent Number
14-17	LHE of Extent (CCHH)
18-21	RHE of Extent (CCHH)
22-25	Last Relative Track Number (HHHH)

LABNAME=Name (Cont'd)

Notes: 1. The size of the extent matrix depends upon two factors: the number of volumes that this file crosses and the number of extents (file areas) on each volume.

2. Bytes 0-12 appear once for each volume.

3. The last 13 bytes must be repeated for each additional extent (up to 15) per volume.

4. The area specified by LABNAME must be large enough to contain all of the extents for all volumes to be opened.

READID=YES

♦ This entry is required if the READID macro is used in the program.

READKEY=YES

♦ This entry is required if the READKEY macro is used in the program.

RECFORM=

♦ This required entry specifies the type of records in the input or output file. One of the following entries must be specified:

FIXUNB - for fixed-length unblocked records. All records are considered unblocked in the direct-access method. If the program requires blocked records, it must perform its own blocking and deblocking.

UNDEF - for undefined records. This entry is required:

1. When records are not fixed length.
2. If an end-of-file record is to be written. (An end-of-file record can be written only by using a WRITEAFTER or WRITEAFTERID macro.)

RECSIZE=n

♦ This entry is required for files containing undefined records (RECFORM= UNDEF). The letter n specifies the number of the general register that contains the length of each individual record. General Registers 2-13 may be used. When undefined records are read FCP supplies the length of the data field of the record in the register. When a record is built, the program must load the length of the data field of the record into the register before the record is written.

RELADDR=YES

♦ This entry must be included if relative track addresses rather than actual track addresses are to be supplied in the track reference field (see SEEKADR).

SEEKADR=Name

♦ This entry is required. Name specifies the symbolic name of the eight-byte track reference field in which the program places the track location of the particular record to be read or written. The specified name must be the same as the name used in the DS instruction that the program must set up for this area. Whenever a record is to be located by searching for a specified ID, the track reference field must also contain the number of the record on the track.

SEEKADR=Name (Cont'd)

Information in the track reference field may be either actual or relative as shown in tables 1-10 and 1-11.

Table 1-10. Actual Track Address Format

Byte Number	Identification	Meaning
0	M	This byte contains the binary volume sequence number on which the record is located. All volumes for a file are numbered consecutively starting with 0. Information concerning volumes is in the extent matrix area at LABNAME. This entry should match the volume sequence number in field 4 of the format 1 label of the VTOC.
1,2	B,B	Reserved for future use.
3	C	If DEVICE=MASS68, this byte contains the actual card number on which the record is located. For all other random access devices, this byte is $(00)_{16}$.
4	C	This byte contains the actual cylinder number in which the record is located.
5	H	This byte is $(00)_{16}$.
6	H	This byte contains the head number (the track within the cylinder) that is used to reference the record.
7	R	This byte contains the sequential number (1-255) of the record on the track (0-255 if WRITE AFTERID is used). R must be provided when a record is to be referred to by ID. If a record is not to be referred to by ID, this byte is ignored.

Table 1-11. Relative Track Address Format

Byte Number	Identification	Meaning
0-2	Not Used	These bytes are reserved for future use.
3-6	TTTT	These bytes contain the binary track number relative to the beginning of the file. The first track is numbered 1.
7	R	This byte contains the sequential number (1-255) of the record on the track (0-255 if WRITE AFTERID is used). R must be provided when a record is to be referred to by ID, otherwise, this byte is ignored.

VERIFY = YES	◆ If records are to be checked after they are written, this entry should be included. Because data written to the 70/568 Mass Storage Unit is automatically checked when it is written, this entry is not applicable if DEVICE = MASS68.
WRITEID = YES	◆ This entry is required if the WRITEID macro is used in the program.
WRITEKY = YES	◆ This entry is required if the WRITEKEY macro is used in the program.

SEEKADR=Name (Cont'd)

Notes:

1. The user must not address the first track of the first extent of each volume. It is used for labels or label information whether labels are present or not.
2. The volume sequence number must be zero if relative track addressing is used.

The volume sequence number must be zero if relative track addressing is used.

SRCHM=YES

◆ If records are to be referred to by key, this optional entry causes FCP to search multiple tracks for each specified record. Execution of the READKEY or WRITEKEY macro causes a search to take place for the specified key beginning with the track specified in the track reference field and all following tracks in the cylinder. The search continues until the record is found or the end of cylinder is detected. If this entry is not included, the search is confined to the track specified in the track reference field.

TYPEFLE=

◆ This entry indicates how user labels are to be processed. One of the following entries can be specified.

INPUT - user labels are to be read and written.

OUTPUT - user labels are to be written.

If this entry is omitted, INPUT is assumed.

USEIN=Name

◆ *User Exit on Initiation.*

This facility provides the programmer with the ability to obtain control from FCP whenever FCP initiates a physical write to the file. Thus the programmer can create a backup file which can be used to facilitate reconstruction of a file.

To generate the User Exit routine the programmer must specify in his DTFDA or DTFSR the keyword USEIN=name. This causes FCP to create a 32-byte communication table (Table 1-5A) and transfer control to the user routine immediately prior to executing a physical write order to the random access device. FCP builds this communication table beginning at the user's USEIN address.

The programmer's USEIN subroutine must be a physical level operation and must maintain the integrity of registers 0, 1,14, and 15. The users returns control back to FCP by use of a BR 14 instruction. FCP then completes the write operation and normal processing continues. No logical FCP macros can be executed within the user routine and the routine must be double word aligned (example: Name DS OD).

Table 1-12. DTFDA Entry Summary Table

Name	Operation	Operand	564 Disc Storage Unit 565 Drum Memory Unit 568 Mass Storage Unit	Remarks
Filename	DTFDA	AFTER = YES	X *	Required if WRITE AFTER macro is used.
		AFTERID = YES	*	Required if WRITE AFTERID macro is used.
		ALTAR = YES	*	Specifies Dynamic Alternate Track Assignment.
		BLKSIZE = n	X	Size of I/O Area.
		CAPREC = YES	*	Required if WRITE AFTER macro is used.
		CONTROL = YES	*	Required if CNTRL macro is used.
		DEVICE = DISC64 DRUM65 MASS68	X	Type of device.
		DLYPRX = name	*	Delayed Processing Facility. For 70/568 Mass Storage Unit only.
		ERRBYTE = Name	X	Area used to store information concerning terminating conditions after an I/O operation.
		IDLOC = Name	*	Specifies an area that FCP stores the ID of the record just processed.
		IOAREA1 = Name	X	Specifies input/output area for this file.
		KEYARG = Name	*	Required if READKEY or WRITEKEY macro is used.
		KEYLEN = n	*	Required if records are to be referenced by key.
		LABADDR = Name	*	Required if additional standard labels are present.
		LABNAME = Name	X	Specifies extent matrix area.
		READID = YES	*	Required if READID macro is used.
		READKEY = YES	*	Required if READKEY macro is used.
		RECFORM = FIXUNB UNDEF	X	Specifies record format.
		RECSIZE = n	*	Required if record format is undefined.
		RELADDR = YES	*	Required if relative track addresses are to be supplied.
		SEEKADR = Name	X	Specifies the address of the track reference field.
		SRCHM = YES	*	Permits searching over multiple tracks.
		USEIN = name	*	User Exit on Initiation.
		TYPEFLE = INPUT OUTPUT	*	If this entry is omitted input is assumed.
		VERIFY = YES	*	Causes records to be verified after they are written (not required for 70/568 Mass Storage Unit).
		WRITEID = YES	*	Required if WRITEID macro is used.
		WRITEKY = YES	*	Required if WRITEKEY macro is used.

Legend: X = required entry * = optional entry.

DTFEN Describe File

GENERAL DESCRIPTION

♦ The DTFEN macro is a descriptive macro which must follow all of the DTF entries for the program. It indicates to the assembler that all files have been described.

FORMAT

♦ The format for DTFEN macro is as follows:

Name	Operation	Operand
	DTFEN	

SPECIFICATION RULES

Name Field

♦ Must be blank.

Operation Field

♦ DTFEN.

Operand Field

♦ Not used.

OPEN
Open File

GENERAL DESCRIPTION

♦ The OPEN macro activates each file that is to be utilized in the program.

FORMAT

♦ The format for the OPEN macro is as follows:

Name	Operation	Operand
	OPEN	Filename
	OPEN	Filename1, Filename2 ...

SPECIFICATION RULES

Name Field

♦ Not required.

Operation Field

♦ OPEN.

Operand Field

♦ This entry is the symbolic name of the logical file assigned in the DTFSR, DTFDA, or DTFPH entry for the file. As many as 16 files can be opened with one OPEN macro by entering additional file names. They are opened in the same order as specified in the macro.

General Specifications

♦ For the card reader, card punches, printer, paper tape reader, and paper tape punch, the OPEN macro simply makes the file available for input or output. When a tape file with standard labels is OPENed, FCP expects the first record read to be a label. However, if other specifications are given, or if a file starting in the middle of the tape is OPENed, it is the program's responsibility to position the tape properly so that the first record read is a label. Positioning is done by the OPEN macro if the DTF entry MRKCTR is specified (or if a FILES run-time parameter is present). If the first record is not a label, FCP indicates an error condition by issuing a message to the operator. An unlabeled file (DTFSR FILABL=NO) can, however, be opened in the middle of data records without causing an error condition. If a file with nonstandard labels (DTFSR FILABL=NSTD) is OPENed, all label processing is the responsibility of the program.

Whenever an input/output tape or a direct-access file is to be OPENed and the program requires additional standard labels or nonstandard labels to be processed, the program must provide the information for checking or building the labels. If this information is obtained from another input file, that file must be opened ahead of the tape or direct-access file. This is done by specifying the input file ahead of the tape or direct-access file in the same OPEN macro, or by issuing a separate OPEN macro preceding the OPEN for the tape or direct-access file.

Input Files (Tape) (Cont'd)

If the compare on fields 3-10 is unequal the user can direct FCP to terminate and dump, retry, type the tape record causing the error and the user supplied label information, or to replace the user supplied information with that in the tape record. Additional HDRn records (up to a maximum of eight) are bypassed. An unrecognizable record following the last HDR1 record causes the FCP to request the user to direct it to terminate and dump, retry, type the record causing the error and the user supplied label information, or to read the next record. In this last case, FCP expects the next record to be recognizable, that is, either a UHL record or a tape mark.

User header labels, those with UHLn in the first four bytes, are bypassed unless the user has specified a LABADDR address in his DTFSR in which case FCP branches to the users LABADDR routine after reading each UHLn label. When the user returns to FCP with the LBRET 1 macro. or after FCP has read a total of eight UHL labels or when FCP encounters a tape mark when attempting to read the next label, standard label checking for the start of this file on this reel is considered finished.

If reverse processing a tape mark must be the first record read.

Nonstandard Labels - FCP branches to the users LABADDR routine. if there is one. The user reads his nonstandard labels and a tape mark using physical level FCP and returns to FCP using the LBRET Δ1 macro. FCP then considers the file opened.

If there is no LABADDR routine FCP bypasses any nonstandard labels preceding the tape mark. A maximum of 24 nonstandard labels is bypassed The operator can then direct FCP to terminate and dump, retry, or search through 24 more records for the tape mark.

If LABADDR has been specified there must be at least one nonstandard label preceding the tape mark.

No Labels - FCP reads the first block. If it is a tape mark FCP considers the file opened. If it is not a tape mark FCP rewinds one gap and considers the file opened.

If the file is to be read reverse, (BACK specified in DTFSR entry READ), the OPEN macro expects to find the file trailer label set. User trailer labels are handled identical to UHL labels on forward processing. Additional EOV or EOF labels, if present, are bypassed. The file trailer label must be complete with block count. The label is checked and the block count is stored to be checked when the file is terminated.

Output Files (Tape)

◆ When an output file is recorded on magnetic tape, the OPEN macro rewinds the tape according to the specifications in the DTFSR entry REWIND. Rewind is not performed if the file is defined by DTFPH.

Output Files (Cont'd)

Standard Labels - FCP reads the first record. If it does not contain VOL1 in the first four bytes the operator can direct FCP to terminate and dump, retry, type the record causing the error and the user-supplied label information, test this record to see if it is a HDR1 record, or to write the HDR1 record. If the write HDR1 record option is selected the VOL1 record is left as is. If the first record is not a VOL1 but is a tape mark, the user can direct FCP to terminate and dump, retry, or to read the next record ib which case the label check procedure is reinitialized. In this case, FCP expects the next record read to be a VOL1 record.

If the first record is a VOL1, FCP compares the volume serial number (label field 3) in the VOL1 record with the volume serial number on the user supplied label information.

If the compare on volume serial number is not equal, the operator can direct FCP to terminate and dump, or retry, or type the record causing the error, or to replace the volume serial number in the user supplied label information with the volume serial number on the VOL1 tape record.

If the compare on volume serial number is equal, FCP reads the next tape record. All VOLn records (up to a total of eight) records are bypassed. The first record following the last VOLn record is tested for HDR1.

If it is not a HDR1 record FCP rewinds one gap and writes a HDR1 label with today's date in field nine (creation date). The rest of this record is set to blanks (40_{16}).

If it is a HDR1 record FCP tests the expiration date (field 10) on this label with today's date (in the executive communication region)entered at system initialization.

If the expiration date in the label is greater than today's date, the operator can direct FCP to terminate and dump, retry, type the record causing the error and the user supplied label information, or to override the label check. In this case FCP rewinds one gap and writes a new header label over the old one. This label contains the user supplied information except that creation date is set equal to today's date. The rest of the label is blanks (40_{16}).

If the purge date is equal to or less than today's date FCP rewinds one gap and writes a new HDR1 record containing the user supplied label information except that today's date is posted in the record as the creation date. The rest of the label is set to blanks (40_{16}).

FCP then writes a tape mark and branches to the user's LABADDR routine, if present. When the user returns to FCP via the LBRET 1 macro or after eight UHLn labels have been written, or if the user has no LABADDR routine, FCP writes a tape mark. This completes the label checking procedures for this file.

Output Files (Cont'd)

Non-standard Labels - FCP branches to the user's LABADDR routine so that the user can write his nonstandard labels. When the user has written all his nonstandard labels, using physical level FCP, he returns to FCP via the LBRET macro. FCP writes a tape mark unless TPMARK=NO has been specified. In either case FCP then considers the file opened.

No Labels - FCP writes a tape mark unless TPMARK=NO has been specified. After writing the tape mark, or immediately if not required, FCP considers the file opened.

Input Files (Tape)

◆ When an input file is recorded on magnetic tape, the OPEN macro rewinds the tape according to the specifications in the DTFSR entry REWIND. If LABNAME is not specified, the OPEN macro finds the label information in the run-time parameter area. Multifile reel users may also use the OPEN macro to position the tape by specifying the number of tape marks to be skipped. (See DTFSR MRKCTR=n.) If LABNAME is specified, the OPEN macro expects the label information to be complete in this area.

Standard Labels - If the file sequence number in the label information indicates that this is the first file on the volume, the OPEN macro expects the first block to be a volume label. If not the first file on the volume then the first record must be HDR1.

FCP reads the first tape record. If it is not a VOL1 record, the user can direct FCP to terminate and dump, retry, or type the tape record causing the user supplied label information. This error cannot be bypassed unless the record is a tape mark, in which case the user can direct FCP to read the next record or to consider the file opened.

If the first record read is a VOL1 record, FCP compares the volume serial number (field 3) on the label with that supplied by the user. If this is the second or a succeeding reel of a multivolume file this check of volume serial number is omitted.

If the volume serial numbers are not equal, the user can direct FCP to terminate and dump, retry, or replace the error volume serial number in his parameters with that in the tape VOL1 record.

If the volume serial numbers are equal FCP reads the next block. All records with VOLn (up to a total of eight) are bypassed. The first record after the last VOLn record is tested for HDR1.

If this is not a HDR1 record, the user can direct FCP to terminate and dump, retry, type the error record and his label information, or to bypass the HDR1 check in which case the next label check is for user labels.

If this record is not a HDR1 record but is a tape mark, the user can direct FCP to terminate and dump, retry, consider the file opened, or to read the next record in which case the label check is reinitialized.

If this record is a HDR1 record FCP compares fields 3-10 of the HDR1 label with fields 3-10 of the user supplied label information (in a single compare operation). If this is the second or a succeeding reel of a multivolume file FCP increments the volume sequence number in the user supplied label information by one before making the compare on fields 3-10.

Direct-Access Files

◆ When a direct-access file is OPENed (input or output file) the OPEN routine refers to the information supplied by the user in Volume Displacement Cards (run-time parameter card). A Volume Displacement Card must be supplied for every volume on which the file to be OPENed is recorded and is to be processed. Volume Displacement Cards are described in the TOS Control System Manual (No. 70-00-609) under Executive Run Time Parameters.

A volume is defined as one of the following:

1 - 70/564 Disc Storage Unit

1 - 70/565 Drum Memory Unit

1 - 70/568 Mass Storage Magazine

When any direct-access file is OPENed, the OPEN routine utilizes the information contained in the Volume Displacement Cards to generate an extent matrix for the file. The extent matrix is generated in the area specified by the DTFSR, DTFDA, or DTFPH entry LABNAME. An extent is a continuous area on a direct-access device, within a volume, where the file is recorded. Up to 16 extents for a single file may be present within a volume. When a direct-access file is OPENed all of the file's extents for each volume described in the Volume Displacement cards are stored in memory.

When a program that accesses random access device(s) is to be run, the following must take place:

1. The volume(s) to be accessed by the program must be placed on-line.

2. The Update On-Line Catalog console routine must be run. This routine accesses each random access volume (via the device list) that is on-line and reads the volume label. The serial number for each volume is extracted from the volume label and placed in the Executive Resident Catalog. For each serial number posted, the device list entry address is also posted.

3. Before loading the program, Volume Displacement Cards (run-time parameters) containing the filename, File ID, and the serial numbers of the volumes on which this file is contained, must be placed in the card reader.

   If the VDC cards contain a matrix parameter (specifying matrix size) a new matrix area is substituted for the matrix area specified in the DTF entry LABNAME. This allows the user to enlarge the file extent matrix at run time. FCP effects this substitution by changing the address in the pointer that is used to indicate the location of the matrix.

Direct-Access Files (Cont'd)

4. When the program is loaded the run-time parameters are loaded and placed in memory following the program.

5. At OPEN time the following occurs:

   a. OPEN accesses the run-time parameter area and matches the filename there with the filename specified by the OPEN macro.

   b. When a match on filename has been found, OPEN picks up the first file identification (44 bytes) and the first volume serial number following the filename in the run-time parameter area and attempts to match the volume serial number with an entry in the on-line catalog. When a match is found, the device list address is placed in the filename's CCB and the I/O is initiated to the device.

Input Files (Direct Access)

♦ When an input file that is recorded on a direct-access device is OPENed, the OPEN routine:

1. Accesses each volume as specified in the VDC's.

2. Makes any additional user-standard labels available to the user for checking if the DTFSR, DTFDA, or DTFPH entry LABADDR is included in the file definition.

3. Locates the area(s) on the direct-access device (extents) where the file is recorded.

4. Makes the file records available for processing.

To perform these functions the OPEN routines refer to the information supplied by the user in his VDC cards. Prior to execution of the program the user has supplied a VDC card for every volume on which the file is recorded and is to be processed.

The DTF type determines how the OPEN routines perform their functions. For serial processing (DTFSR and DTFPH) OPEN initially checks the standard labels on the first volume; makes the additional labels, if any, available to the user for checking; and then locates and makes available the first extent on the first volume. As the user issues GET's in his program the FCP processes one extent at a time, in the sequence specified in the VDC cards.

When FCP detects the end of the current extent, EOF logic then locates the next extent and makes it available for processing. If the next extent for the file is the first extent of the next volume, EOV logic checks the standard label on the current volume and makes any additional user trailer labels available for checking. Then FCP automatically OPENs the next volume by checking the standard label, making any user additional labels available for checking, and making the first extent on the volume available for issuing GET's.

Input Files (Direct Access) (Cont'd)

For direct-access processing (DTFDA) OPEN processes user header labels as in serial processing. However, this process is repeated at the initial OPEN for all volumes of the file. In other words all header labels for all volumes must be checked during the initial OPEN. Additional user trailer labels are not checked when DTFDA has been specified.

Initial Access. OPEN initially accesses the standard volume label (CCHH=zeros) on the first volume of the file. This label contains the address of the Volume Table of Contents (VTOC). FCP reads in VTOC. The OPEN logic searches the VTOC for the 44-byte file identification specified in the VDC card for this file/volume. When it is found all of the extents for this filename on this volume are brought into memory and stored in LABNAME (a DTF entry). OPEN then repeats the above for all volumes of the file so that the file extent matrix is complete after the initial OPEN for that file.

Checking Additional User-Standard Labels. The programmer can check additional user-standard labels, if any, if he has specified LABADDR in his DTF macro. The OPEN routine, after checking the standard volume label, reads the first user label into IOAREA1 and loads the address of this area into General Register 1. FCP then branches to the user's LABADDR routine. The user can identify the various labels by testing bytes 1-4 of the label. After checking each label the user returns to FCP by issuing the LBRET macro (operand 2). When the user has checked all his labels or has checked as many as desired, the rest to be ignored, he issues the LBRET macro with operand 1. As a maximum of eight additional labels are allowed, FCP automatically terminates label checking after the eighth label is checked.

Each time the user returns to FCP with LBRET, FCP writes the label back onto the device, therefore, the user can update his labels in addition to checking them.

If the user has not specified LABADDR but does have additional labels on the volume, these labels are ignored.

The user is not required to have additional user header or trailer labels even though he may have specified LABADDR in his DTF. When LABADDR is specified FCP returns control to the user's instruction after the OPEN.

Output Files (Direct Access)

◆ When an output file is to be recorded on a direct-access device, the OPEN routine:

1. Creates and writes the user header and user trailer label set. All volumes of the file must be on line when the file is opened.
2. Permits the user to create additional user standard labels if the DTFSR, DTFDA, or DTFPH entry LABADDR is included in the file definition.
3. Locates the area(s) on the direct-access device (extents) where the file is to be recorded.
4. Makes the areas available for processing.

Output Files (Direct Access) (Cont'd)

To perform these functions the OPEN routines refer to the information supplied by the user in his VDC cards. For creation of the standard labels OPEN also uses information supplied by the DTF.

The general procedure is the same for output files as for input files. For serial processing (DTFSR, DTFPH) only the first volume is OPENed at the initial OPEN. After the first volume, each other volume is OPENed and the additional standard user labels are written when, during execution, the processing of records for one volume is completed and the file is to be continued on a different volume. For random processing (DTFDA) all volumes are OPENed and all standard labels are written at the initial OPEN.

After the standard labels are written for a volume, FCP branches to the user's LABADDR routine, if the user has specified one in his DTF. (If LABADDR is not specified no additional user labels are written.) In his LABADDR routine the user constructs his additional labels. OPEN has loaded General Register 1 with the address of IOAREA1. This is where the user places each label that he wants to write. OPEN has also placed UHL1 (for the first label) in the first four bytes of this area. The user builds a 76-byte label and returns to FCP by use of the LBRET macro. FCP writes the label and, if the user has used LBRET operand 1 or if this was the eighth label, terminates the label set. If this was not the eighth label and if the user has used LBRET operand 2, FCP adds one to the label identifier and returns to the user's LABADDR routine.

Whenever LABADDR has been specified at least one additional label must be written. FCP writes an end-of-file record following the last label.

Nonstandard Labels

◆ *Input Files (Tape Only)* - When an input tape file contains nonstandard labels as indicated by the DTFSR entry FILABL=NSTD, the OPEN macro branches to the address in LABADDR if specified. If LABADDR is not specified, the OPEN macro assumes there is a tape mark following the labels that the user does not want checked and positions the file following this tape mark.

In the routine specified by LABADDR, the user must use physical FCP to read and check his labels. When he has read and checked all his labels and positioned the tape for the first GET macro, he must return to the OPEN macro with the LBRET 2 macro.

Note: Files may be processed in reverse.

Output Files (Tape Only) - When an output tape file contains nonstandard labels as indicated by the DTFSR entry FILABL=NSTD, the DTFSR entry LABADDR *must* be specified.

Nonstandard Labels (Cont'd)

When the file is opened, the OPEN macro branches to the address specified by LABADDR with the letter O in General Register 0. In the LABADDR routine, the user must build and write his nonstandard labels. He must issue his own physical FCP commands to accomplish this. Following the writing of his last label, the user returns to the OPEN macro with the LBRET 2 macro. The OPEN macro then writes a tape mark unless DTFSR TPMARK=NO is specified.

Omitted Labels

◆ *Input Files (Tape Only)* - When an input tape file is defined as having labels omitted as indicated by the DTFSR entry FILABL=NO, the OPEN macro reads the first block and if this is a tape mark, the tape is considered positioned at the first data block. If it is other than a tape mark, the tape is repositioned to its original point.

Note: Omitted label files may be processed in reverse.

Output Files (Tape Only) - When an output tape file is defined as having labels omitted as indicated by the DTFSR entry FILABL=NO, the OPEN macro writes a tape mark unless DTFSR entry TPMARK=NO is specified. This will destroy any previous data on this volume including the VOL label.

LBRET Label Return

GENERAL DESCRIPTION

♦ The LBRET macro applies only to tape or direct-access files that have additional user-standard labels, or nonstandard labels (tape files only) that the user wants to check or write. It must be issued at the end of the user label routine (specified by the DTF entry LABADDR), to return to FCP after header or trailer labels have been processed.

FORMAT

♦ The format of the LBRET macro is as follows:

Name	Operation	Operand
	LBRET	1
		2

SPECIFICATION RULES

Name Field

♦ Not required.

Operation Field

♦ LBRET.

Operand Field

♦ Specifications for this entry are described below.

Additional User Standard Labels

♦ *Input Files (Tape and Direct Access)* - Operand "1" specifies a return to FCP when the program wants to eliminate the checking of one or more additional user-standard labels. FCP then skips the remaining labels in the set and processing continues. If all labels are to be checked, operand "1" is not used and FCP terminates label processing when the tape mark or EOF record following the last label is read.

Operand "2" specifies a return to FCP after each additional standard label has been checked. FCP makes the next label available for checking in IOAREA1. If FCP reads the tape mark or after 8 user labels it terminates label processing.

If FCP has read an EOV label it processes labels on the next input reel. If an EOF label has been read FCP branches to the user's EOFADDR.

Output Files (Tape and Direct Access) - Operand "1" specifies a return to FCP when the program determines that the last additional user-standard label has been built. FCP writes the last label (from IOAREA1) and a tape mark (tape files) or EOF record (direct-access files) and processing continues. Operand "1" is always required to terminate the output label set unless the maximum of eight labels is to be written. Label processing is automatically terminated after the eighth label is written.

Operand "2" specifies a return to FCP after each user header or trailer label except the last has been built. FCP writes the label from IOAREA1 and returns to the user's label routine to permit him to build his next label.

Nonstandard Labels

◆ *Input and Output Files (Tape Only)* - Operand "2" is invalid for tape files that contain nonstandard labels. Operand "1" causes a return to FCP after all nonstandard labels have been checked or written. FCP branches to the user's label routine only once for each label set. In the label routine, the program must read or write every required label and must position the tape for receipt or transmission of the first data record before issuing LBRET to return to FCP.

Note: Upon entry to the user's label routine, FCP stores a return address in General Register 14. The LBRET macro uses this address and if the program also requires General Register 14, it must save and restore the contents before issuing the LBRET macro instruction. The program is also responsible for maintaining the contents of General Register 15.

CLOSE
Close

GENERAL DESCRIPTION

♦ The CLOSE macro is used to deactivate any file that was previously opened on any input/output unit in the system.

FORMAT

♦ The format of the CLOSE macro is as follows:

Name	Operation	Operand
	CLOSE	Filename
	CLOSE	Filename1, Filename2 ...

SPECIFICATION RULES

Name Field

♦ Not required.

Operation Field

♦ CLOSE.

Operand Field

♦ The Filename entry is the symbolic name of the logical file assigned in the DTF entry for the file. As many as 16 files may be closed by one instruction by entering additional file names. They are closed in the same order as specified in the macro.

General Specifications

♦ The CLOSE macro is required whenever logical FCP macro instructions are used to transfer data. If physical FCP is used, the CLOSE macro is required only when standard labels are to be checked or written. A file may be closed at any time by issuing the CLOSE macro. If further processing of a closed file is required at some later time in the program, the file must be opened again. However, after a CLOSE for a tape file, the tape is positioned in accordance with the REWIND specification. Therefore, to resume processing of tape records at the point where the CLOSE occurred, NORWD should be specified in DTFSR entry REWIND. When an OPEN macro 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 non standard labels, it is the program's responsibility to identify the first record read as a data record or a file label.

Tape Input File

♦ The tape is rewound as specified in the DTFSR entry REWIND. No labels are read or checked. In all cases label checking for input files is done either in the user's LABADDR (if specified) or the user's EOFADDR.

Tape Output File

♦ For a magnetic tape output file, the CLOSE macro 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 DTFSR entry REWIND.

Tape Output File (Cont'd)

When standard labels are specified (STD in DTFSR entry FILABL or OUTPUT in DTFPH entry TYPEFLE), the CLOSE macro 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.

When logical FCP (DTFSR) is used for an output file, FCP accumulates the block count for the trailer label. When physical FCP (DTFPH) is used, however, the program must accumulate the block count, if desired, and supply it to FCP for inclusion in the standard trailer label. For this, the count must be moved to the four-byte packed decimal field labeled Filename B. In this case, Filename must be seven characters long. For example, if the file name specified in the DTFPH header entry is DETLOUT, the block count field is addressed by DETLOUTB. The user coding must insure that the block count field is an addressable tag.

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

If additional labels are to follow the standard trailer, the CLOSE macro branches to the user's routine (identified by the DTF entry LABADDR) after the standard label has been written. In this routine, the program can build a maximum of eight additional labels, which the CLOSE macro writes for him. After building each user-standard label, the program must return to the CLOSE macro by use of the LBRET macro. After all trailer labels are written, the CLOSE macro writes two tape marks, executes the rewind option, and deactivates the file. For the proper procedures to handle additional user-standard labels and/or nonstandard labels, see Tape Output File description for the OPEN macro.

When nonstandard labels are specified (NSTD in DTFSR entry FILABL) the CLOSE macro causes FCP to give control to the user's LABADDR routine after writing the tape mark following the last data block. The user writes all of his nonstandard labels by use of physical level I/O macros and then returns to FCP by use of the LBRET macro.

Other Files

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

When a printer, or card output file is completed, the CLOSE macro 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.

Direct-Access Input File

♦ When a direct access file is being processed randomly (specified by DTFDA), the CLOSE macro is issued in the problem program to deactivate the file after all records have been processed.

When records in a direct-access input file are processed in serial order (DTFSR), the CLOSE macro is issued in the user's end-of-file routine (EOFADDR) to deactivate the file. FCP branches to this routine when it detects an end-of-file condition.

Direct-Access Output File

♦ When records in a direct-access output file are processed in serial order (DTFSR), the CLOSE macro is issued after all records for the file have been processed. In this case, the CLOSE macro causes one or more functions to be performed before it deactivates the file. FCP always writes an end-of-file trailer record after the last data record in the file. If records are processed in serial order, user trailer labels may be written if the DTFSR entry LABADDR is included in the file definition.

Up to eight user trailer labels can be written on the first track of the first extent specified for the file on each volume. They follow the additional user header labels for the volume and are identified by UTL1, UTL2, ... UTL8. For this operation, FCP branches to the user's label routine, sets up a label area (IOAREA1), and supplies the address of the area in General Register 1.

In his routine the user constructs the trailer label in the area referenced by General Register 1 and then returns control to FCP by use of the LBRET macro. Similar to writing user header labels, these steps are repeated until eight trailer labels have been written or until the user indicates that he does not require any more labels, whichever occurs first. After the last trailer label is written, the CLOSE macro deactivates the file.

The CLOSE macro generates coding that creates linkage to the routine that will output a partially filled block of records, write the correct label records, if any, and mark the file closed.

GET
Obtain Record

GENERAL DESCRIPTION

♦ The GET macro makes the next consecutive 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. When the GET macro detects an end-of-file condition, FCP branches to the user's end-of-file routine (specified by the DTFSR entry EOFADDR) after all data has been processed.

FORMAT

♦ The format for the GET macro is as follows:

Name	Operation	Operand
	GET	Filename
	GET	Filename,Workname

SPECIFICATION RULES

Name Field

♦ Not required.

Operation Field

♦ GET.

Operand Field

♦ Filename is the symbolic name of the logical file assigned in the DTFSR entry for the file. Workname is the symbolic name of the user's work area.

General Specifications

♦ The GET macro may be written in either of two formats, depending on the area where the records will be processed. One form or the other, but not both, can be used for one DTFSR-specified logical file.

First Format

♦ The first format is used if records are to be processed directly in the input area(s). Only one parameter is required. This parameter specifies the name of the file from which the record is to be retrieved. The file name must be the same as the one specified in the DTFSR header entry for this file.

The input area must be specified in the DTFSR entry IOAREA1. Two input areas may be used to permit an overlap of data transfer and processing operations. The second area is specified in the DTFSR entry IOAREA2. Whenever two input areas are specified, the FCP routines transfer records alternately to each area. FCP completely processes this transfer so that the next consecutive record is always available to the user's program for processing.

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

1. Records are blocked, or

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

First Format (Cont'd)

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

Second Format

♦ The second format is used if records are to be processed in a work area. It causes the GET macro to move each individual record from the DTFSR-specified input area to a work area. As in the first format, the filename must be entered as the first parameter. The name of the work area must be entered as the second parameter, and YES must be specified in the DTFSR entry WORKA. The work-area name must be the same as that specified in the instruction that reserves this area of memory. All records 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 macro for the file specifies the same work area. In the second case, different GET macro statements specify different work areas. It may be advantageous to plan two work areas and to specify each area in alternate GET macros. This permits the programmer to compare each record with the preceding one for a control change. Only one work area can be specified in any single GET macro. Whenever this format of the GET macro is used for a logical file, a register may not be used for indexing (as it is when records are processed directly in the input area).

The size of the work area is the record size. The size of the I/O area(s) is the block size.

Unblocked Records

♦ Records retrieved from any input file (except for direct-access or magnetic tape files) are always considered unblocked (specified as unblocked or undefined). Records on direct-access or magnetic tape devices are treated as unblocked if this is specified in the DTFSR entry RECFORM. Whenever records are unblocked (either fixed- or variable-length), and only one input area is used, each GET macro 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 macro. If two input areas are specified, each GET makes the last record that was transferred to memory available for processing in the input area or work area. The same GET macro also starts the transfer of the following record to the other input area.

For variable-length unblocked records the block length and record length fields are both brought into memory preceding the data.

Blocked Records

♦ When records on direct-access or magnetic tape devices are specified as blocked in the DTFSR 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 macro transfers a block of records from direct-access 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 macros 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.

Undefined Records

♦ When undefined records are to be processed, the DTFSR entries RECFORM=UNDEF and RECSIZE=n must be included in the file definition. The GET macro treats undefined records as unblocked, and the programmer must locate individual records and fields. Undefined records are considered to be variable in length by FCP because no other characteristics of the record are known by FCP.

Read Backwards

♦ If records are to be read backwards (BACK specified in DTFSR entry READ), blocks of records, or unblocked records, are transferred from tape to memory in reverse order. The last block is read first, the next-to-the-last-block, second, etc. For blocked records, each GET macro 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 unblocked variable length records are to be processed, a register must be specified in the DTFSR entry IOREG. FCP stores the starting location of the record in the specified register.

Note: The above applies only to 9-channel tapes. Variable-length records created on a 7-channel tape cannot be read backwards. Fixed-length blocked or unblocked records can only be read reverse on a 7-channel tape if:

1. Translate on, pack/unpack off.
2. Translate off, pack/unpack off.

Fixed-length records may be read reverse with pack/unpack on only if the record length is a multiple of three.

Updating

♦ A consecutive file on a direct-access unit can be updated. That is, each direct-access record can be read, processed, and transferred back to the same direct-access location, from which it was read. This function must be specified by the DTFSR entry UPDATE=YES. The direct-access record is transferred to memory by a GET macro instruction. After the record is processed, the next PUT macro instruction causes the updated record to be written in the same location from which the record was read. PUT transfers the record to the direct-access file from the *input* area of memory. If a work area is specified in the GET and PUT macros, PUT first moves the updated record from the work area back to the input area and then transfers the record to this file. For a file to be updated the TYPEFLE=INPUT entry must appear in the DTFSR.

A GET macro must always precede a PUT instruction for a direct-access record and only one PUT macro can be issued for each record. A PUT macro can be omitted if a particular record does not require updating.

PUT
Output Record

GENERAL DESCRIPTION

♦ The PUT macro writes, punches, or displays logical records that have been built directly in the output area or a specified work area. It is used for any output file in the system and for any type of record. The macro is similar to GET, but is issued after a record has been built:

FORMAT

♦ The format for the PUT macro is as follows:

Name	Operation	Operand
	PUT	Filename
	PUT	Filename,Workname

SPECIFICATION RULES

Name Field

♦ Not required.

Operation Field

♦ PUT.

Operand Field

♦ Filename is the symbolic name of the logical file assigned in the DTFSR entry for the file. Workname is the symbolic name of the work area specified in the DS instruction that reserves this area of memory.

General Specifications

♦ Similar to GET, the PUT macro is written in either of two formats, depending upon the area where the records are built. One form or the other, but not both, can be used for one DTFSR-specified logical file.

First Format

♦ The first format is used if records are built directly in the output area(s). Only one parameter is required and this parameter specifies the name of the file to which the record is to be transferred. The filename must be the same as the one specified in the DTFSR header entry for this file.

The output area must be specified in the DTFSR entry IOAREA1. Two output areas may be used to permit an overlap of data transfer and processing operations. The second area is specified in DTFSR entry IOAREA2. Whenever two output areas are specified, the FCP routines transfer records alternately from each area. FCP completely processes this transfer so that the proper output record area is always available to the program for the next consecutive output record.

When records are built in the output area(s), a general register must be specified in the DTFSR 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. The PUT macro places the proper address in the register.

Second Format

♦ The second format is used if records are built in a work area. It causes the PUT macro to move a record from a specified work area to the proper location in the DTFSR-specified output area. As in the first form, the filename must be entered as the first parameter. The name of the work area is entered as the second parameter, and YES must be specified in the DTFSR entry WORKA. The work area name must be the same as that specified in the DS instruction that reserves the area of memory. Individual records for a logical file may be built in the same work area or in different work areas. Each PUT macro specifies the work area where the completed record was built. Only one work area can be specified in any one PUT macro. Whenever this form of the PUT macro is used for a logical file, a register may not be used for indexing.

Unblocked Records

♦ Records transferred to any output file (except for direct-access or magnetic tape devices) are always considered unblocked (specified as unblocked or undefined). Records for direct-access or magnetic tape devices are treated as unblocked if this is specified in the DTFSR 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 macro, the record is first moved from the work area to the output area (or input area) and then to the file.

If RECFORM=VARUNB, the user must place record length (RL) in output area + 4 if one I/O area is used. If two I/O areas are used, the user places the record according to the contents of IOREG.

Blocked Records

♦ When blocked records are to be written on magnetic tape or direct-access devices (as specified in DTFSR entry RECFORM), the individually built records must be formed into a block in the output area. The block of records is then 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 macro for these records either adds an indexing factor to the register, 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, the PUT macro 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 user's program and included in the output record as it is built. The user's 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 read variable-length input 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 macro performs approximately the same functions as it does for fixed-length blocked records. The PUT macro checks the length of each output record to determine if the record fits in the remaining portion of the output area. If the record fits, PUT immediately moves the record. If it does not fit, PUT causes the completed block to be written and then moves the record. This record then becomes the first record in a new block.

Tape End of Reel Logic for Output Files

◆ (FCP has detected the end-of-tape marker during a PUT)

Standard Labels:

1. FCP writes a tape mark.
2. FCP writes a EOV label.
3. FCP branches to user's LABADDR, if there is one.

   If there is no LABADDR, or after the user issues LBRET 1 in his LABADDR routine, FCP writes a double tape mark.
4. FCP rewinds and unloads the tape.

   If ALTTAPE is specified FCP requests a device assignment from the Executive, processes labels on the next reel, and returns to the user at the instruction following the PUT.

   If ALTTAPE is not specified, FCP issues a request for the operator to dismount this reel and mount the next reel. When the operator replies to this message FCP processes labels on the next reel and returns to the user at the instruction following the PUT.

Nonstandard Labels:

1. FCP writes a tape mark.
2. FCP branches to the user's LABADDR routine.

   When the user returns to FCP via the LBRET 2 macro, FCP writes a double tape mark. FCP rewinds and unloads the reel.

   If ALTTAPE is specified FCP requests a device assignment from the Executive and branches to the user's LABADDR routine. When the user returns to FCP via the LBRET 2 macro, FCP returns to the user at the instruction following the PUT.

   If ALTTAPE is not specified FCP returns to the user's LABADDR routine. A message is typed requesting the operator to swap reels. The user returns to FCP once again via the LBRET 2 macro and FCP returns to the user at the instruction following the PUT.

No Labels:

1. FCP writes a double tape mark.
2. FCP rewinds and unloads the tape.

   If ALTTAPE is specified, FCP requests a device assignment from the Executive, writes a tape mark, (unless TPMARK=NO in the DTFSR) and returns to the user at the instruction following the PUT.

   If ALTTAPE is not specified, FCP issues a request for the operator to dismount the reel and mount the next reel. When the operator replies to this message FCP writes a tape mark (unless TPMARK=NO in the DTFSR), and returns to the user at the instruction following the PUT.

Blocked Records (Cont'd)

If variable-length blocked records are to be built directly in the output area, an additional DTFSR entry VARBLD, a 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 beinning of a new block. Thus, the length of the record must be calculated and compared with the amount of area remaining.

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

Undefined Records

♦ When undefined records are processed, PUT treats them as unblocked. The programmer must provide any desired blocking. The length of each record (in bytes) must be determined and loaded into a register for FCP use before the PUT macro is issued for that record. The register that will be used for this purpose must be specified in the DTFSR entry RECSIZE.

Updating

♦ Refer to GET macro.

Printer Control Considerations

♦ Line spacing or skipping in a printer can be controlled either by specified characters in the data records or by the CNTRL macro. One method or the other, but not both, may be used for a particular logical file.

When control characters in data records are to be used, the DTFSR entry CTLCHR must be specified, and every record must contain a control character. This must be the *first* character of each fixed-length record, or the first character following the record length in a variable-length record. The particular character included in the record is determined by the function to be performed. For example, if double spacing is to occur before a particular line is printed, the code for double spacing must be the control character in the output line to be printed.

If the CNTRL macro is used for nondata orders to the printer, the DTFSR entry CONTROL is specified and the DTFSR entry CTLCHR must be omitted. In this case, any control characters included in data records are treated as data when the PUT macro is executed.

RELSE Release

GENERAL DESCRIPTION

♦ The RELSE macro is used with blocked input records read from magnetic tape or direct-access devices. It allows the program to skip the remaining records in a block, and to continue processing with the first record of the next block when the next GET macro is issued.

FORMAT

♦ The format for the RELSE macro is as follows:

Name	Operation	Operand
	RELSE	Filename

SPECIFICATION RULES

Name Field

♦ Not required.

Operation Field

♦ RELSE.

Operand Field

♦ Filename is the symbolic name of the logical file specified in the DTFSR entry for the file.

This function can be used in a job where records on tape are categorized and each category (perhaps a major grouping) is planned to start with a new block. For selective reports, specified categories can be located readily by checking only the first record in each block.

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

TRUNC
Truncate

GENERAL DESCRIPTION

♦ The TRUNC macro is used with blocked output records that are written on tape or direct-access devices. It allows the program to write a short block of records. When the end of a category of records is reached, that block is written, and the new category is started at the beginning of a new block.

FORMAT

♦ The format of the TRUNC macro is as follows:

Name	Operation	Operand
	TRUNC	Filename

SPECIFICATION RULES

Name Field

♦ Not required.

Operation Field

♦ TRUNC.

Operand Field

♦ Filename is the symbolic name of the logical file specified in the DTFSR entry for the file.

When the TRUNC macro is issued, the block is written and the output area is made available to build the next block. The last record included in the written block is the record that was built before the last PUT macro was executed. Therefore, if records are built in a work area and the user's program determines that a record belongs in a new block, the TRUNC macro should be issued first, followed by the PUT macro for this particular record. If records are built in the output area, however, the program must determine if a record belongs in the block before it builds the record.

Whenever variable-length blocked records are built directly in the output area, the TRUNC macro must be used to write a completed block of records. When the PUT macro is issued after each variable-length record is built, the output routines supply the program with the space (number of bytes) remaining in the output area. From this information the program determines if his next variable-length record fits in the block. If it does not fit, the TRUNC macro is issued 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 by the DTFSR entry VARBLD.

CNTRL Control

GENERAL DESCRIPTION

◆ The CNTRL macro provides the program with the means of specifying physical nondata operations for magnetic tape units, direct access, and printers. Such functions as rewinding tape, and line spacing are provided. When a CNTRL macro is executed, FCP does not wait for completion of the macro before returning control to the program.

FORMAT

◆ The format of the CNTRL macro is as follows:

Name	Operation	Operand
	CNTRL	Filename,Code[,[n][,m]]
	CNTRL	Filename,SEEKSR

SPECIFICATION RULES

Name Field

◆ Not required.

Operation Field

◆ CNTRL.

Operand Field (First Format)

◆ The first format is used for all nonrandom access I/O devices. The Filename entry is the symbolic name of the logical file specified in the DTFSR entry for the file. The code entry is the mnemonic code which specifies the control function to be performed. The acceptable codes are given in table 1-13.

Table 1-13. Control Function Mnemonic Codes

Mnemonic Code	Function	I/O Device
REW	Rewind tape.	Magnetic Tape
RUN	Rewind and unload.	Magnetic Tape
BSR	Backspace to interrecord gap.	Magnetic Tape
BSF	Backspace to tape mark.	Magnetic Tape
FSR	Forward space to interrecord gap.	Magnetic Tape
FSF	Forward space to tape mark.	Magnetic Tape
WTM	Write tape mark.	Magnetic Tape
ERG	Erase gap (writes blank tape).	Magnetic Tape
SP	Carriage space n or m lines.*	Printers
SK	Skip to channel n or m.	Printers

*The n and m entries are used for printer control operations. The n entry is for immediate operations. The m entry is for delayed operations. Immediate and delayed operations cannot both be specified simultaneously. The m and n entries represent the number of lines to be spaced or the channel in the carriage control loop to which the paper is to be advanced (skipped).

Examples:

To space three lines immediately:

```
CNTRL    filename,SP,3
```

To space three lines after the next print command (delayed):

```
CNTRL    filename,SP,,3
```

To skip to channel one immediately:

```
CNTRL    filename,SK,1
```

To skip to channel one after the next print command (delayed):

```
CNTRL    filename,SK,,1
```

Operand Field (First Format) *(Cont'd)*

Whenever CNTRL is issued in the user's program, the DTFSR entry CONTROL must be included in the file definition. The CNTRL macro must not be used for printer files if the data records contain control characters and the DTFSR entry CTLCHR is included in the file definition.

Magnetic Tape Devices

♦ The following factors should be considered with input tape movement operations (BSR, BSF, FSR, and FSF):

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

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

3. If blocked input records are being processed and the program does not want to process the remaining logical records in the block or one or more succeeding blocks (physical records), he must issue a RELSE macro before issuing the CNTRL macro. Then the next GET makes the first record of the new block available for processing. If the CNTRL macro, with FSR for example, is issued without a preceding RELSE, the tape advances, but the next GET makes 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, FCP 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 is already in memory. Therefore, if a CNTRL FSR operation is performed, the second physical tape record beyond the present one is passed without being read into memory.

5. If an FSR operation (or BSR in a read backwards file) passes a tape mark, FCP branches to the end-of-volume routine.

6. The CNTRL macro cannot be used for the Paper Tape Reader/Punch.

7. If any of these four functions is used during the processing of a file, the block count accumulated for checking standard labels, or accumulated for the checkpoint macro, may be wrong. The operator can bypass an erroneous block count when checking the standard labels. However, since it is impossible to reposition the tape correctly if the block count is wrong in a checkpoint record, these commands should not be issued when using the checkpoint macro.

Printers

♦ The CNTRL macro is used for any printer forms control other than the standard single spacing. The CNTRL macro codes for printer operations 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), and the fourth parameter 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.

Second Format (Direct-Access Devices)

♦ For direct-access devices, the CNTRL macro can be used 1) to terminate logical processing on the current block and 2) to initialize the address of the next block as determined by the information pointed to by the DTFSR entry SEEKADR.

Note: The CNTRL macro can be used only when one IOAREA has been specified.

Input Mode - When the user issues the CNTRL macro in the input mode, FCP releases the current block being processed and reads the block pointed to by the DTFSR entry SEEKADR. The next GET macro to this file supplies the first record in this block.

Output Mode - The CNTRL macro, when issued in the output mode, causes FCP to: 1) truncate the current block being processed, 2) write this block out to the address already set up by previous writes, and 3) set up the next write to the address specified by the DTFSR entry SEEKADR.

Update Mode - In the update mode, FCP performs the following: 1) causes the current block to be released, 2) writes in the update mode this block if PUT'S have been issued to it; and 3) reads the block specified by the DTFSR entry SEEKADR. The next GET to this file supplies the first record in the block.

PRTOV Printer Overflow

GENERAL DESCRIPTION

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

FORMAT

◆ The format of the PRTOV macro is as follows:

Name	Operation	Operand
	PRTOV	Filename,n,Routine-name

SPECIFICATION RULES

Name Field

◆ Not required.

Operation Field

◆ PRTOV.

Operand Field

◆ The Filename entry is the symbolic name of the logical file assigned in the DTFSR entry for the file.

The n entry specifies the number of the carriage tape channel (9 or 12) used to indicate the overflow. This is entered as 9 or 12. When these two entries are specified, and an overflow condition occurs, and Routine-name has not been specified, FCP automatically restores the printer carriage to the first printing line on the form (channel 1), and printing of detail lines continues.

The Routine-name optional entry is entered in this instruction 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. This entry specifies the symbolic name of the user's routine. In this case, FCP does not restore the carriage to channel 1. The user routine may issue any macro instructions (except PRTOV) to perform whatever functions are desired. For example, the programmer can print total lines, skip to channel 1, and print overflow headings. At the end of the routine the user must branch back to FCP by way of the address stored by FCP in General Register 14. Therefore, if any FCP macros are used in the routine the user must save the addresses that are in General Registers 14 and 15 and restore them when he has finished using the FCP macros.

Note:

1. After an overflow condition is detected one more line is printed before going to channel 1.

FEOV Forced End of Volume

GENERAL DESCRIPTION

♦ The FEOV macro is used for either input or output files on tape to force an end-of-volume condition when neither a tape mark nor an end-of-tape warning marker has been sensed. 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 to the next volume.

FORMAT

♦ The format of the FEOV macro is as follows:

Name	Operation	Operand
	FEOV	Filename
	FEOV	Filename1,Filename2...

SPECIFICATION RULES

Name Field

♦ Not required.

Operation Field

♦ FEOV.

Operand Field

♦ The Filename entry is the symbolic name of the logical file assigned in the DTF entry for the file.

When logical FCP macros are used for a file (DTFSR specified), the FEOV macro initiates the same functions that occur at a normal end-of-volume condition, except for trailer-label checking.

For an input tape, the FEOV macro immediately rewinds the tape as specified by the DTFSR entry REWIND and provides for a volume change as specified by the DTFSR entry ALTTAPE. Trailer labels are not checked. The FEOV macro then checks the standard header label on the new volume, and provides for program checking of any additional standard header labels if the DTFSR entry LABADDR is specified. If nonstandard labels are specified (DTFSR entry FILABL=NSTD), FEOV macro provides for program checking if desired.

For an output tape, FEOV macro writes the last block of records, if necessary (this may be a short block), and writes a tape mark. Then the FEOV macro writes the standard trailer label and additional user standard labels (if any), writes two tape marks, provides for a volume change, and writes the file header label(s) on the new volume, as specified in the DTFSR entries REWIND, ALTTAPE, FILABL, and LABADDR. If nonstandard labels are specified, the FEOV macro provides for the user-writing of trailer labels (completed volume) and header labels (new volume), if desired.

When physical FCP macro instructions are used and DTFPH is specified for standard label processing FEOV may be used for input and output files.

Operand Field *(Cont'd)*

For input files (physical level processing), FCP expects that the user has read a tape mark following his data and that the tape is currently positioned prior to the trailer label. The FEOV macro then processes the trailer label, and determines if it is an end-of-volume or an end-of-file condition. If an end-of-volume condition is detected, FCP then processes the header labels of the new volume, positions the tape beyond the tape mark, and returns to the user with General Register 1 containing the characters EV in the low-order bytes. If an end-of-file condition is detected, FCP returns to the user with General Register 1 containing the characters EF in the low-order bytes. For this case, however, since General Register 1 is destroyed the macro must be issued singularly or the DTFPH must be the last filename entered in the macro.

For output files (physical level processing), FCP writes the tape marks, the trailer label, and two tape marks on the original volume; purge checks the next volume; writes header label and tape mark; and returns to the user.

In each of the above FEOV cases, input and output, additional user label processing is satisfied through the normal use of the LABADDR entry.

READ
Read
Record

GENERAL DESCRIPTION

◆ The READ macro causes a physical record to be read from a random access device and placed in the memory area specified by IOAREA1 (DTFDA).

FORMAT

◆ The READ macro can be written in either of two formats as follows:

Name	Operation	Operand
	READ	Filename,KEY
	READ	Filename,ID

SPECIFICATION RULES

Name Field

◆ Not Used.

Operation Field

◆ READ.

Operand Field

◆ Filename is the symbolic name of the logical file assigned in the DTFDA entry for this file. KEY specifies that the type of reference used to search for a record is the KEY. ID specifies that the type of reference used to search for a record is the ID (Record number). Both forms can be used to reference the same file if the file has keys.

Reference by Key

◆ If the READ KEY macro is to be executed the DTFDA entries READKEY, KEYLEN, and KEYARG must be present. Prior to the execution of a READ KEY macro, the program must supply the following information:

1. The key for the record to be read must be stored in the area specified by the KEYARG entry in the DTFDA.

2. The address of the track to be searched (actual or relative) must be stored in the track reference field (bytes 0-6) specified by the SEEKADR entry in the DTFDA.

3. It is the user's responsibility to restore the record length general register if the user program alters it.

When the READ KEY macro is executed FCP does the following:

1. The key field of the first record encountered on the specified track is compared with the key field supplied by the program.

2. If the keys are equal, the data field of the record is transferred to IOAREA1. The key is not brought into memory.

3. If the keys are not equal, the key field of the next record on the track is compared with the key field supplied by the program. This

Reference by Key (Cont'd)

continues until a key is found that matches or one of the following occurs:

a. The entire track is searched and the specified key is not found. Then "no record found" bit in the error/status byte is set to 1.

b. If the DTFDA entry SRCHM=YES is included, the specified track and all following tracks in the cylinder are searched. If the specified key is not found and end of cylinder is detected, the "end of cylinder" bit in the error/status byte is set to 1.

Reference by ID

♦ If the READ ID macro is to be executed, the DTFDA entry READID must be present. Prior to the execution of a READ ID macro, the program must supply the following information:

1. The ID (record number 1-255) must be stored in the track reference field (byte 7) specified by the SEEKADR entry in the DTFDA.
2. The address of the track to be searched (actual or relative) must be stored in the track reference field (bytes 0-6) specified by the SEEKADR entry in the DTFDA.

When the READ ID macro is executed, FCP does the following:

1. The ID of the first record on the specified track is compared with the ID supplied by the program.
2. If the ID's are equal one of the following occurs:
   a. If KEYLEN is specified in the DTFDA, both the key and the data fields of the record are transferred to IOAREA1 + 8.
   b. If KEYLEN is not specified in the DTFDA, only the data field of the record is transferred to IOAREA1 + 8 + key length.
3. If the ID's are not equal, the ID of the next record on the track is compared with the ID supplied by the program. This continues until a match is found or an end-of-track condition is detected. If an end-of-track condition is detected before a match is found, the "no record found" bit in the error/status byte is set to 1.

Note: When a READ ID macro is executed, only the specified track is searched even if the SRCHM entry is specified in the DTFDA.

Notes

1. If a READ macro is executed and the number of bytes requested is less than the number of bytes specified in the count field of the record (key length-KL, data length-DL) only the number of bytes requested are transferred to IOAREA1.
2. If a READ macro is executed and the number of bytes requested is greater than the number of bytes specified in the count field of the record (key length-KL, data length-DL) the number of bytes specified in the count field of the record is transferred to IOAREA1. In this case, the "wrong length record" bit in the error/status byte is set to 1.
3. The contents of IOAREA1 are explained more fully in the section on the CNTRL macro (direct access).

WRITE
Write
Record

GENERAL DESCRIPTION

♦ The WRITE macro causes a physical record to be written from the memory area specified by IOAREA1 (DTFDA) to a random access device.

FORMAT

♦ The WRITE macro can be written in four formats. Two of the formats are used to update records already existing on the random access device. The other two formats are used to add new records to the file. The formats of the WRITE macro are as follows:

Name	Operation	Operand
	WRITE	Filename,KEY
	WRITE	Filename,ID
	WRITE	Filename,AFTERID
	WRITE	Filename,AFTER

SPECIFICATION RULES

Name Field

♦ Not required.

Operation Field

♦ WRITE.

Operand Field

♦ Filename is the symbolic name of the logical file assigned in the DTFDA entry for this file. The KEY operand is used when updating an existing record on the file. KEY specifies that the type of reference used to search for a record is the KEY. The ID operand is used when updating an existing record on the file. It specifies that the type of reference used to search for a record is the ID (Record Number). The AFTERID operand is used when a new record is to be added to the file. AFTERID specifies that a specific record (referred to by record number) is to be searched and that the new record is to be added after this record. The AFTER operand is used when a new record is to be added to the file. AFTER specifies that the record is to be added after the last record on the track.

Record Reference by Key

♦ The WRITE KEY macro is used to update the data field of an *existing record* on a random access device. If this macro is to be executed, the DTFDA entries WRITEKY, KEYLEN, and KEYARG must be present. Prior to the execution of a WRITE KEY macro the program must supply the following information:

1. The key for the record to be written must be stored in the area specified by the KEYARG entry in the DTFDA.
2. The address of the track to be searched (actual or relative) must be stored in the track reference field (bytes 0-6) specified by the SEEKADR entry in the DTFDA.
3. The data field to be written must be placed in IOAREA1.

Record Reference by Key (Cont'd)

When the WRITE KEY macro is executed, FCP does the following:

1. The key field of the first record encountered on the track is compared with the key field supplied by the program.

2. If the keys are equal, the data field contained in IOAREA1 is transferred to the data field of the record on the random access device. The number of bytes transferred to the data field of the record is determined by the data length (DL) specified in the count field of the record. If the number of bytes in IOAREA1 is less than the data length as specified in the count field, the remainder of the data field of the record is filled with 0's. If the number of bytes in IOAREA1 is greater than the data length as specified in the count field, only the number of bytes as specified in the count field is transferred. In this case, the "wrong length record" bit in the error/status byte is set to 1 when the WAITF macro is executed.

3. If the keys are not equal, the key field of the next record on the track is compared with the key field supplied by the program. This continues until a key is found that matches or one of the following occurs:

   a. The entire track is searched and the specified key is not found. The "no record found" bit in the error/status byte is set to 1.

   b. If the DTFDA entry SRCHM is included, the specified track and all following tracks in the cylinder are searched. If the specified key is not found and an end-of-cylinder condition is detected, the "end-of-cylinder" bit in the error/status byte is set to 1.

Record Reference by ID

♦ The WRITE ID macro is used to update the key and/or data field of an *existing record* on a random access device. If this macro is to be executed, the DTFDA entry WRITEID must be included. Prior to the execution of a WRITE ID macro, the program must supply the following information:

1. The ID (record number 1-255) must be stored in the track references field (byte 7) specified by the SEEKADR entry in the DTFDA.

2. The address of the track to be searched (actual or relative) must be stored in the track reference field (bytes 0-6) specified by the SEEKADR entry in the DTFDA.

3. The key (if present) and data fields to be written must be placed in IOAREA1.

When the WRITE ID macro is executed, FCP does the following:

1. The ID of the first record on the specified track is compared with the ID supplied by the program.

Record Reference by ID (Cont'd)

2. If the ID's are equal one of the following occurs:

   a. If KEYLEN is specified in the DTFDA, the key and data fields contained in IOAREA1 are transferred to the key and data fields of the record on the random access device.

   b. If KEYLEN is not specified in the DTFDA, the data field contained in IOAREA1 is transferred to the data field of the record on the random access device.

   c. The number of bytes transferred is determined by the key length (KL) and data length (DL) specified in the count field of the record. If the number of bytes in IOAREA1 is less than the number of bytes specified by the count field, the remainder of the record is filled with 0's. If the number of bytes in IOAREA1 is greater than the number of bytes specified by the count field, only the number of bytes as specified in the count field is transferred. In this case, the "wrong length record" bit in the error/status byte is set to 1 when the WAITF macro is executed.

3. If the ID's are not equal, the ID of the next record on the track is compared with the ID supplied by the program. This continues until a match is found or an end-of-track condition is detected. If an end-of-track is detected before a match is found, the "no record found" bit in the error/status byte is set to 1.

Note: When a WRITE ID macro is executed, only the specified track is searched even if the SRCHM entry is specified in the DTFDA.

Record Reference by AFTERID

♦ The WRITE AFTERID macro is used to add a new record after a record containing a specified ID (record number) on a track. If this macro is to be executed, the DTFDA entry AFTERID must be present. Prior to the execution of a WRITE AFTERID macro the program must supply the following information:

1. The ID (record number 0-255) must be stored in the track reference field (byte 7) specified by the SEEKADR entry in the DTFDA. The ID to be specified is the ID of the record after which the new record is to be written.

2. The address of the track to be searched (actual or relative) must be stored in the track reference field (bytes 0-6) specified by the SEEKADR entry in the DTFDA.

3. The key (if present) and data fields to be written must be placed in IOAREA1.

When the WRITE AFTERID macro is executed, FCP does the following:

1. The ID of the first record on the specified track is compared with the ID supplied by the program.

Record Reference by AFTERID (Cont'd)

2. If the IDs are equal, one of the following occurs:

   a. If CAPREC is specified in the DTFDA, FCP checks the capacity record to see if the new record fits on the track. If it fits, the new record (count field, key field if present, and data field) is transferred from IOAREA1 to the track immediately following the record with the specified ID and the remainder of the track is erased. The capacity record is then updated to reflect the addition. If the capacity record indicated that the new record does not fit on the track, the record is not written and the "no-room found" bit in the error/status byte is set to 1.

   b. If CAPREC is not specified in the DTFDA, FCP transfers the new record (count field, key field if present, and data field) from IOAREA1 to the random access device immediately following the record with the specified ID and the remainder of the track is erased. If there is not enough room on the track for this record only that part of the record that fits is written and a track overrun condition occurs. The "track-overrun" bit in the error/status byte is set to 1.

   Note: The count field to be written is always generated by FCP.

Record Reference by AFTER

♦ The WRITE AFTER macro is used to add a new record after the last existing record on a track. If this macro is to be executed the DTFDA entries AFTER and CAPREC must be present. Prior to the execution of a WRITE AFTER macro, the program must supply the following information:

1. The address of the track on which the record is to be added (actual or relative) must be stored in the track reference field (bytes 0-6) specified by the SEEKADR entry in the DTFDA.
2. The key (if present) and data field must be stored in IOAREA1.

When the WRITE AFTER macro is executed, FCP does the following:

1. FCP checks the capacity record on the track to determine the location and the amount of space available for the record.
2. If the new record fits, it is transferred from IOAREA1 (count field, key field if present, and data field) to the track immediately following the last record and the remainder of the track is erased. The capacity record is then updated to reflect the addition.
3. If the capacity record indicated that the new record does not fit on the track, the record is not written and the "no-room-found" bit in the error/status byte is set to 1.

   Notes: 1. The WRITE AFTER macro can only be used with files containing valid capacity records.

   2. The count field to be written is always generated by FCP.

Notes:

♦ 1. If records in the file are specified as undefined (DTFDA entry RECFORM = UNDEF), the program must supply the length of the data field only of the record in the register specified by the DTFDA entry RECSIZE. If a key field is to be written in addition to the data field, the length of the key field is not included. FCP determines the key length from the KEYLEN entry in the DTFDA.

2. The contents of IOAREA1 are explained more fully in the section on the CNTRL macro (direct access).

WAITF
Wait

GENERAL DESCRIPTION

♦ The WAITF macro is issued when the program requires that a previously initiated Read or Write macro be completed before execution of the program can continue.

FORMAT

♦ The format of the WAITF macro is as follows:

Name	Operation	Operand
	WAITF	Filename

SPECIFICATION RULES

Name Field

♦ Not required.

Operation Field

♦ WAITF.

Operand Field

♦ Filename is the symbolic name of the logical file assigned in the DTFDA to which the Read or Write macro was directed.

When this macro is executed the program is placed in a wait status until the Read or Write operation is completed. When the operation has been completed, control is returned to the program with information stored in the error/status bytes. A WAITF macro must be issued for each Read or Write operation because the information contained in the error/status bytes is available only upon issuance of this macro.

CNTRL Control

GENERAL DESCRIPTION

◆ The CNTRL macro is issued when the program wants to overlap seek operations (physical positioning of a random access device) with processing.

FORMAT

◆ The format of the CNTRL macro is as follows:

Name	Operation	Operand
	CNTRL	filename,SEEK

SPECIFICATION RULES

Name Field

◆ Not required.

Operation Field

◆ CNTRL.

Operand Field

◆ Filename is the symbolic name of the logical file assigned in the DTFDA entry for this file. SEEK is a keyword that is required.

General Specifications

◆ Prior to execution of this macro the program must supply information in the first seven bytes (0-6) of the track reference field (actual or relative). The track reference field is specified by the SEEKADR entry in the DTFDA.

When this macro is issued, FCP issues a SEEK command. As soon as the SEEK command has been initiated, control is transferred to the program. It is not necessary to issue a WAITF macro after a CNTRL macro because the next I/O operation issued to the device causes a check of the SEEK to be made. If another I/O operation is issued to the same device while a SEEK is in progress, a waiting loop is automatically entered until the SEEK is complete. I/O operations to other files on other devices can be issued while a SEEK is in progress.

Page 1-84 was deleted by
August 1967 revision.

Logical Processing Summary Table

Macro Instructions	Applies to							
	70/221-10-11-20-21 Paper Tape Reader Punch	70/234-10-11 70/236-10-11-20-21 Punch Units	70/237-10-21-22 Reader	70/242-10-20 70/243-10-20 70/248-10-11 Printers	70/432-1-2 70/442-1-2 70/445-1-2 Magnetic Tape Devices	70/564 Disc Storage Unit	70/565 Drum Memory Unit	70/568-1-2 Mass Storage Unit
Serial + Direct Access Macros								
OPEN	X	X	X	X	X	X	X	X
LBRET					X	X	X	X
CLOSE	X	X	X	X	X	X	X	X
Serial Only Macros								
GET	X	X	X	X	X	X	X	X
PUT	X	X		X	X	X	X	X
RELSE	X				X			
TRUNC	X				X			
CNTRL				X	X	X	X	X
PRTOV				X				
FEOV				X				
Direct Access Only Macros								
READ						X	X	X
WRITE						X	X	X
WAITF						X	X	X
CNTRL						X	X	X

PROGRAMMING I/O USING PHYSICAL LEVEL FCP

GENERAL

♦ Physical level FCP provides a means of transferring data to and from I/O devices and of performing nondata operations on I/O devices. Physical FCP macros are available to provide direct communication with the I/O routines contained in the Executive. Physical FCP also provides standard label checking routines for tape and direct-access files. Logical data handling functions that require interrogation of the characteristics of a data file are not provided by Physical FCP routines.

The Physical Level FCP macros and a brief statement of their use are as follows:

File Descriptor Macros

♦ DTFPH - describes the file when using physical level FCP.

DTFEN* - Must allow all DTF macros and indicates to the assembler that all files have been described.

I/O Control Macros

♦ OPEN* - Activates a file in the user's program.

LBRET* - Provides the ability to check or write additional standard labels or nonstandard labels.

CLOSE* - Detects a file that was previously opened.

FEOV* - Forces an end-of-volume condition.

*These macros are identical to the macros described under Logical Level FCP, File Descriptor and I/O Control Macro Section.

CCB** - creates control information required by the Executive to issue an I/O operation. The CCB also receives status information after completion of an I/O operation.

EXCP** - is issued to request that an I/O operation be performed.

EXCPW** - is issued to request that an I/O operation be performed and that the calling program be "waited" until the I/O operation has completed.

WAIT** - is issued when the program requires that a previously initiated I/O operation be completed before continuing.

CHECK** - is used to check the status of an I/O operation.

CPCI** - is used to determine if a program controlled interrupt has occurred during an I/O operation.

**The I/O Control macros are also considered Executive Communication Macros.

CCW Instruction

♦ CCW - defines the I/O operation to be performed.

MACRO FORMAT

♦ The macro format for the control macros used in programming I/O using physical level FCP is the same as described under logical level FCP.

DTFPH Define the File by Physical FCP

GENERAL DESCRIPTION

◆ The DTFPH macro is used to define files when the program desires to do its own input/output by using physical I/O control macros. Only those files that require standard label checking or writing procedures need be defined. No other files require definition. Because standard labels are required for random access devices, it is recommended that this macro be included in order to use physical I/O control macros to address random access files.

FORMAT

◆ The format for the DTFPH macro is given in table 1-14.

Table 1-14. Format of DTFPH Macro

Name (1-7 Characters)	Operation	Operand
Filename	DTFPH	ALTTAPE=nnnnnn, DEVADDR=nnnnnn, DEVICE={DISC64 / DRUM65 / MASS68}, ISRKEY=n, LABADDR=Name, LABNAME=Name, MRKCTR=nnnn, OVRTN=Name, REWIND={UNLOAD / NORWD}, TYPEFLE={INPUT / OUTPUT}

SPECIFICATION RULES

Name Field

◆ This is a required entry and contains the filename.

Operation Field

◆ DTFPH.

Operand Field

◆ Each operand entry shown in table 1-14 is described below.

ALTTAPE=nnnnnn

◆ This entry specifies the symbolic name for a magnetic tape device that is used as an alternate when a tape file is specified in the DTFPH entry DEVADDR=nnnnnn. *This is an optional entry for tape files.*

DEVADDR=nnnnnn

◆ This entry specifies the symbolic device name to be associated with this tape file. *This entry is not applicable for direct-access files.* The symbolic device name must be unique for each logical file and is used by the Executive Control System to assign the actual I/O device to this file. Whenever two devices are used for one logical file (specified in DTFPH ALTTAPE), this entry specifies the symbolic device name for the first device.

DEVICE=

◆ This entry is required if the I/O device type associated with this logical file is a random access device. One of the following must be entered for random access files:

DISC64 - for an input and/or output file on the 70/564 Disc Storage Unit.

DRUM65- for an input and/or output file on the 70/565 Drum Memory

MASS68 - for an input and/or output file on the 70/568 Mass Storage Unit.

Notes:

1. If this entry is omitted, magnetic tape is assumed.
2. Only tape and random access devices are handled by the DTFPH.

ISRKEY=n

◆ This entry must be included for direct-access files if keys are to be read or written. The letter n specifies the number of bytes (1-256) in each key. All keys must be the same length. If this entry is omitted a key length of zero is assumed. This entry is used by FCP to determine the location of the data field in IOAREA1. The ISRKEY = n entry is for serial processing only.

LABADDR=Name

◆ This entry is used when the program requires one or more labels in addition to the standard file header or trailer label. Name indicates the symbolic name of the program's subroutine that checks or builds the additional labels. FCP branches to this subroutine after it has processed the standard label. At the end of this subroutine, the program must return to FCP by using the LBRET macro. The processing of labels by FCP is described below.

Input Files - Additional Standard Header Labels (Tape and Direct Access).

1. FCP branches to this routine after it has checked the standard volume and header labels.
2. When control is received by the routine, General Register 1 contains the address of where the additional user header label is located.
3. The program must determine that a header label requires processing as opposed to a trailer label.
4. The program must return to FCP after processing each additional header label by using the LBRET macro.

LABADDR=Name (Cont'd)

5. Upon completion of routine processing (all additional header labels have been processed), FCP positions the file past the tape mark (tape) or the EOF record (direct-access).

6. If additional user standard header labels are present and the LABADDR entry is not included for the file, FCP bypasses these labels and positions the file past the tape mark (tape) or the EOF record (direct-access).

Input Files - Additional Standard Trailer Labels (Tape and Direct Access)

1. Upon detection of a tape mark (tape) or an EOF record (direct access), the program must transfer control to FCP via FEOV macro.

2. When control is received by the LABADDR routine, General Register 1 contains the address of where the additional user trailer label is located.

3. The program must determine that a trailer label requires processing as opposed to a header label. On trailer label processing the program is responsible for determining end-of-volume or end-of-file condition. In either case, the program must return to FCP after processing each label by using the LBRET macro.

4. Upon completion of the LABADDR routine processing on an end-of-volume or an end-of-file condition, FCP returns control to the program at the instruction following the FEOV macro.

5. When control is returned to the program, General Register 1 contains the following:

General Register 1=

Not Used		E	V, F
Byte	Byte	Byte	Byte

EV = End-of-volume
EF = End-of-file

6. If additional user standard trailer labels are present and the DTFPH entry LABADDR is not included for the file, FCP ignores the additional labels.

Output Files - Additional Standard Header Labels (Tape and Direct Access)

1. FCP branches to this routine after it has checked the volume label, checked the old standard header label, and has written the new standard header label.

2. When control is received by the LABADDR routine, General Registers 0 and 1 contain the following information:

LABADDR=Name (Cont'd)

General Register 0 =

Not Used			O
Byte	Byte	Byte	Byte

Letter O indicates an additional standard header label is to be written.

General Register 1 = LHE address of where the additional standard header label is to be built.

3. The program is responsible for building the entire label in the area specified by General Register 1 each time control is received.

4. The program returns to FCP after writing each additional header label by using the LBRET macro.

5. Upon completion of routine processing (all additional header labels have been written), FCP writes a tape mark (tape) or an EOF record (direct access) following the last additional header label.

Output Files - Additional Standard Trailer Labels (Tape and Direct Access)

1. FCP branches to this routine after it has written the standard trailer labels on either an end-of-volume or end-of-file condition.

2. An end-of-volume condition occurs when the program issues a FEOV macro.

3. An end-of-file condition occurs when the program issues a CLOSE macro for the file.

4. When control is received by the LABADDR routine General Registers 0 and 1 contains the following information:

General Register 0 =

Not Used			V or F
Byte	Byte	Byte	Byte

V = End-of-volume
F = End-of-file

General Register 1 = LHE address of where the additional trailer label is to be built.

5. The LABADDR routine is responsible for building the entire label in the area specified by General Register 1 each time it receives control.

6. The routine returns to FCP after writing each additional trailer label by using the LBRET macro.

LABADDR=Name (Cont'd)

General Consideration - When a LABADDR routine is specified for a file (input or output), the routine is responsible for both header and trailer label processing associated with that file. The routine receives control from FCP at this entrance location for both header and trailer label processing. The routine, in turn, must then determine which type of label requires processing and take appropriate action accordingly.

LABNAME=Name

◆ For tape files, this entry indicates the location of the file's volume and standard labels. For direct-access files, this entry indicates the location of the area where FCP builds the file's extent matrix. For *tape files* and *direct-access files* this is an optional entry.

1. *Tape Files* - this entry applies only to files containing standard labels. It indicates to FCP the area where the file's standard label information is located. From the information contained in this area, FCP checks the file's volume label and checks or builds the file's standard header and trailer labels. When this entry is used, the information must be available in the area prior to the OPENing of the file.

   The format of the standard label information area is as follows:

Bytes	Contents
0-6	Filename-one to seven character alphanumeric field which contains the name of the file.
7-75	For input files, fields 3 to 10 of the file header label. For output files, fields 3 to 13 of the file header information. Unused positions should be set to $(40)_{16}$.
76-81	Reserved for use by FCP.

   If this entry is omitted the program must enter label information during program initiation by using run-time label parameter entries (VOL and TPLAB). During the OPENing of the file, FCP locates the label information for this file and inserts the applicable address into the LABNAME location of the DTFSR expansion.

   Note: When the LABNAME entry is included for the file, run-time label parameter entries may not be entered for the file.

2. *Direct-Access Files* - LABNAME points to the area where FCP builds the file's extent matrix. The area must provide sufficient memory space to hold the address matrices for as many extents as required by the file.

   The user has the ability, however, to describe or override the size of the matrix area at object time.

   If the LABNAME entry is omitted, a run-time parameter VDC statement must be entered defining the size of the matrix.

LABNAME=Name (Cont'd)

If the LABNAME entry is used, the user still has the ability at object time to enter run-time parameter VDC statements to override the LABNAME matrix size.

The format of the extent matrix is indicated below. This format is repeated for each volume.

Format of Extent Matrix

Bytes	Contents
0-5	Volume Serial Number
6-7	Assignment Halfword
8-9	Bin Number (Binary)
10	Volume Sequence Number (Binary)
11	Number of Bytes in this Entry
12	Last Extent Indicator
13-25	*Extent Description No. 1*
13	Extent Number
14-17	LHE of Extent (CCHH)
18-21	RHE of Extent (CCHH)
22-25	Last Relative Track Number (HHHH)

Notes: The last 13 bytes must be repeated for each additional extent (up to 15) per volume.

The size of the extent matrix depends upon two factors: the number of volumes that this file crosses and the number of extents (file areas) on each volume.

MRKCTR=nnnn

♦ This is an *optional* tape entry and, if *omitted,* FCP assumes the file is positioned properly when the file is OPENed. If used, nnnn is a one to four character numeric field which indicates the number of tape marks the tape is to be forward spaced. The positioning takes place during the OPENing of the file.

OVRTN=Name

♦ This optional entry for tape and direct-access files specifies the symbolic name of the program's routine to which FCP transfers control following completion of its end-of-volume logic. At the time control is received, FCP has processed the header labels for the files' next volume.

OVRTN=Name (Cont'd)

Within the routine the program may not issue any FCP macros. Control is returned to FCP by transferring to the location specified in General Register 14.

If this routine is used to control checkpoint dumping, the "transfer control at restart time" as specified in the CKPT macro should stay in the same path as the macro itself. That is, the program must return from the OVRTN routine by using General Register 14.

REWIND=

◆ If this optional entry is not used, tape files are automatically rewound, but not unloaded on an OPEN or CLOSE macro and an end-of-volume condition. If other operations are desired, one of the following operands must be specified:

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.

TYPEFLE=

◆ This entry must be included to specify the type of file (input or output). One of the following entries must be specified.

INPUT - must be specified for magnetic tape or direct-access input files.

OUTPUT - must be specified for magnetic tape or direct-access output files.

DTFPH Entry Summary Table

Name	Operation	Operand	70/432-1, 2 70/442-1, 2 70/445-1, 2 Magnetic Tape	70/564 Disc Storage Unit	70/565 Drum Memory Unit	70/568 Mass Storage Unit
Filename	DTFPH	ALTTAPE=	*			
		DEVADDR=	X	X	X	X
		DEVICE=		X	X	X
		LABADDR=	*	*	*	*
		LABNAME=	*	*	*	*
		MRKCTR=	*			
		OVRTN=	*	*	*	*
		TYPEFLE=	X	X	X	X

X = Required entry * = Optional entry

CCB Command Control Block

GENERAL DESCRIPTION

◆ The CCB macro causes a 40-byte Command Control Block to be created. This block is required to communicate information to the Executive so that a physical I/O operation can be initiated. The Command Control Block also receives status information after the input/output operation has terminated.

FORMAT

◆ The format of the CCB macro is as follows:

Name	Operation	Operand
Blockname	CCB	nnnnnn, command list name [,X'yy',device class] [,dxc-routine]

SPECIFICATION RULES

Name Field

◆ The blockname entry is required and is the symbolic name associated with the first byte of the Command Control Block. This name may be either a unique CCB name or the DTF name (filename) and is required in the EXCPW, EXCP, WAIT, CHECK, and CPCI macros.

Operation Field

◆ CCB.

Operand Field

◆ The nnnnnn entry is the symbolic device name for the actual I/O device with which this CCB is associated.

The command list name must be the symbolic name of the first Channel Command Word (CCW) instruction to be used with this CCB. X'YY' is an optional self-defining term that becomes the user flag byte in the CCB. The device class entry is an optional self-defining term and becomes the device class byte in the CCB if written. The device classes are specified, as shown in table 1-15. The DXC routine is applicable to TDOS only, and is the address of the user's DXC routine.

Note:

There are three sense bytes and all are set to zero at the beginning of the EXCP or EXCPW macro.

Operand Field *(Cont'd)*

Table 1-15. Device Classes

Hex Code	Device Classes
$(00)_{16}$	Magnetic Tape
$(01)_{16}$	Printer
$(02)_{16}$	Bill Feed Printer
$(03)_{16}$	Card Punch 70/234
$(04)_{16}$	Card Punch 70/236
$(05)_{16}$	Card Reader
$(06)_{16}$	Paper Tape Reader/Punch
$(07)_{16}$	Disc Storage Unit
$(08)_{16}$	Drum Memory Unit
$(09)_{16}$	Mass Storage Unit
$(0A)_{16}$	Videoscan Document Reader
$(FF)_{16}$	Accept Assignment of any Device Class

CCB

◆ The Command Control Block generated by the CCB macro is shown in table 1-16.

Table 1-16. Command Control Block

Byte	Bit Position	Title
0-5		Symbolic Device Name.
6		Device Class.
7		User Flag Byte.
	0-1	Reserved - must be 0's. 00 = SDV 01 = TDV
	2	Avoid Device Error Recovery.
	3	Accept Unrecoverable Error.
	4	Optional File
	5	Avoid Direct-Access Error Recovery Messages.
	6	0 - Physical I/O, 1 - Logical I/O
	7	Reserved - must be 0.
8-11		Address of Channel Command Word (CCW).
12-13		Reserved.
14-15		Assignment Halfword.
16		Device Number (from CAR).
17-19		Address of Next CCW (from CAR).

CCB (Cont'd)

Table 1-16. Command Control Block *(Cont'd)*

Byte	Bit Position	Title
20		Flags (from CCR-II).
	0-4	Flags.
	5-7	Zeros.
21		Channel Status Byte (from CCR-II).
22-23		Remaining Byte Count (from CCR-II).
24		Command Code (from CCR-I).
	0-3	Zeros.
	4-7	Command Code.
25-27		Data Address of first byte or location of new CCW if command is TIC (from CCR-I).
28-30		Data Bytes - 70/55 only (from Assembly/ Status Register).
31		Standard Device Byte (from Assembly/Status Register).
32-34		Device Sense Bytes.
35		Executive Flag Byte.
	0	Termination (initially = 1).
	1	Abnormal Condition at Termination.
	2	Unrecoverable Error.
	3	Program Controlled Interrupt.
	4-5	Condition Code.
	6	Sense Information Lost.
	7	No File.
36-39		Channel Address Register if PCI occurs.
40-43		User DXC routine. TDOS only.
44-47		General Register 10 stored on DXC interrupt. TDOS only.
48-51		General Register 11 stored on DXC interrupt. TDOS only.

Bytes 0-5 of CCB

◆ Contains the symbolic device name as specified in the first operand of the CCB macro.

Byte 6 of CCB

◆ Contains the device class as specified in the fourth operand of the CCB macro. (See table 1-15.)

Byte 7 of CCB

◆ Contains the user flag byte as specified in the third operand of the CCB macro. If the third operand is not specified, this byte is assembled as X'00'. The bit significance of this byte is shown in table 1-17.

Byte 7 of CCB (Cont'd)

Table 1-17. Bit Significance of Byte 7 of CCB

Bit Position	Title	Meaning
0-1	Reserved	These bits are set to 0's at assembly time regardless of the operand specified in the CCB macro.
2	Avoid Device Error Recovery	This bit is set by the program: 1. If set to 0 and a device error occurs, standard error recovery is performed. If standard error recovery is successful, processing continues and the program is not notified of the error. If standard error recovery is unsuccessful, an error message is typed to the operator. 2. If set to 1 and a device error occurs, standard error recovery is not performed and the Accept Unrecoverable Error bit (bit 3 of byte 7) in the CCB is tested.
3	Accept Unrecoverable Error	This bit is set by the program. If standard error recovery was performed and was unsuccessful, an error message is typed to the operator. If the operator uses a reply of 1, this bit is tested and: 1. If set to 0 the job is terminated. 2. If set to 1 control is transferred to the program. If standard error recovery was not performed, this bit is tested and: 1. If set to 0 the job is terminated. 2. If set to 1 control is transferred to the program.
4	Optional File	This bit is set by the program: 1. If set to 0 a device must be assigned to this file. 2. If set to 1, it indicates that this file need not be present in order to continue. At device assignment time, the operator can type in a special reply indicating that no device is to be assigned. If the operator types this reply and this bit is set to 1, no device is assigned to this file, the no file bit in Exec flag byte is set to 1, and processing continues.

Byte 7 of CCB (Cont'd)

Table 1-17. Bit Significance of Byte 7 of CCB *(Cont'd)*

Bit Position	Title	Meaning
5	Avoid Direct Access Error Recovery Messages	This bit is set by the program: 1. If set to 0 and standard error recovery is performed and is unsuccessful, an error message is typed to the operator (see Accept Unrecoverable Error-bit 3). 2. If set to 1 and standard error recovery is performed and is unsuccessful, error messages to the operator are bypassed and the job is terminated or control is transferred to the program depending on the setting of the Accept Unrecoverable Error-bit 3.
6-7	Reserved	These bits are set to 0's at assembly time regardless of the operand specified in the CCB Macro.

Bytes 8-11 of CCB

◆ Contain the address of the CCW associated with the CCB as specified by the second operand of the CCB macro.

Bytes 12-13 of CCB

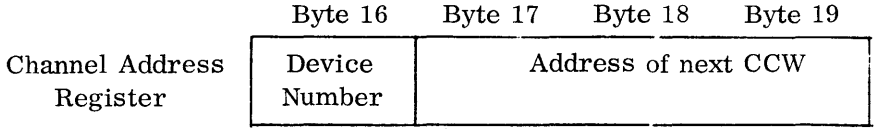
◆ Reserved.

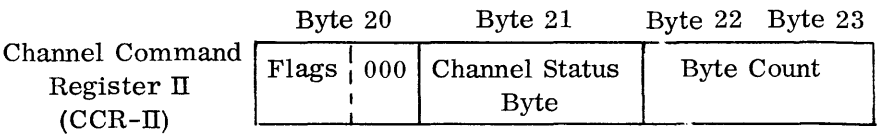
Bytes 14-15 of CCB

◆ Assignment Halfword. When a file is assigned, the address of the device list entry for the device assigned to this file is placed here.

Bytes 16-31 of CCB

◆ Contain the information contained in the channel registers in scratchpad memory at the termination of an I/O operation. The format and contents of the channel registers are as follows:





Bytes 16-31 of CCB (Cont'd)

	Byte 24	Byte 25	Byte 26	Byte 27
Channel Command Register I (CCR-I)	0000 | Command Code	Data Address of First Byte or Location of new CCW if Command is a Transfer in Channel.		

	Byte 28	Byte 29	Byte 30	Byte 31
Assembly/Status Register	Data Bytes (70/55 only)			
				Standard Device Byte

Note: For a detailed description of the contents of these registers refer to the 70/35-45-55 Processors Reference Manual.

Bytes 32-34 of CCB

♦ Contain the device sense bytes. The first sense byte is in byte 32, the second in byte 33, and the third in byte 34. These bytes are assembled as 0's and are set to 0's when an EXCP macro is executed.

Byte 35 of CCB

♦ Contains the Executive flag byte. The bit significance of this byte is shown in table 1-18.

Table 1-18. Bit Significance of Byte 35 of CCB

Bit Position	Title	Meaning
0	Termination	1. Set to 0 when an EXCP or EXCPW macro is executed. 2. Set to 1 when the I/O operation initiated by the EXCP or EXCPW has terminated (successfully or unsuccessfully). Error recovery, if permitted, is performed.
1	Abnormal Condition at Termination	1. Set to 0 when an EXCP or EXCPW macro is executed. 2. Set to 1 when the termination bit (bit 0) is set to 1 and an abnormal condition is indicated at the termination of the I/O operation. Table 1-19 indicates, by device, the conditions that cause this bit to be set.
2	Unrecoverable Error	1. Set to 0 when an EXCP or EXCPW macro is executed.

(Cont'd)

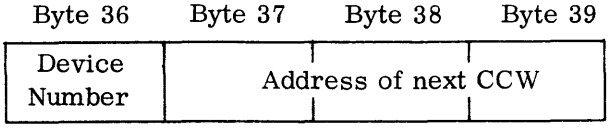
Byte 35 of CCB (Cont'd)

Table 1-18. Bit Significance of Byte 35 of CCB (Cont'd)

Bit Position	Title	Meaning
2 (Cont'd)	Unrecoverable Error (Cont'd)	2. Set to 1 when the termination bit (bit 0) is set to 1 and: a. A device error is detected and standard error recovery is not performed. b. A device error is detected and standard error recovery is performed but is unsuccessful. Table 1-19 indicates, by device, the conditions that cause this bit to be set.
3	Program Controlled Interrupt	1. Set to 0 when an EXCP or EXCPW macro is executed. 2. Set to 1 if a program controlled interrupt occurs during an I/O operation. Note: This bit can be tested by using the CPCI macro.
4,5	Condition Code	1. Set to 0 when an EXCP or EXCPW macro is executed. 2. The condition code is stored in these bits if the I/O could not be initiated. (For a detailed description of the condition code settings, refer to the 70/35-45-55 Processors Reference Manual.)
6	Sense Information Lost	1. Set to 0 when an EXCP or EXCPW macro is executed. 2. Set to 1 if a Sense operation is executed and the sense data is lost.
7	No File	Set to 1 at device assignment time if the Optional File bit is set and the operator types in a special reply indicating that no device is to be assigned to this file.

Bytes 36-39 of CCB

♦ Contain the information contained in the Channel Address Register in scratch pad memory if a Program Controlled Interrupt (PCI) occurs. The format and contents of the Channel Address Register are as follows:



Note: If the CCB is for a card punch, the first four bytes following the CCB should be the absolute address of a backup area. This area is used by error recovery.

Bytes 40-43 of CCB

♦ The dxc-routine is the symbolic name of the DXC routine supplied by the user. The dxc-routine entry is supported under TDOS only.

Table 1-19. I/O Sense Byte Condition (Nondirect-Access Devices)

Device \ Bit Position Set	2^7	2^6	2^5	2^4	2^3	2^2	2^1	2^0
70/236 70/234 Card Punch	Punch Comparison Error (PCE) ■	Punch Memory Parity Error ■	Not Used	Trans-mission Parity Error ■	Intervention, Required, Stacker full, Hopper empty, Chip box full ■	Hold (See Note 2) ■	Not Used	Illegal Operation ⊠
70/237 Card Reader	Read Error ■	Service Request Not Honored ■	Stacker Selection Too Late ⧄	Invalid Punch Code ■	Not Used	Hold (See Note 2) ■	Not Used	Illegal Operation ⊠
70/221 Paper Tape Reader	Read Parity Error ■	Service Request Not Honored ■	Not Used	Not Used	Intervention Required ■	Hold (See Note 2) ■	Not Used	Illegal Operation ⊠
Punch	Punch Parity Error ■	Not Used	Not Used	Not Used	Intervention Required ■	Hold (See Note 2) ■	Low Tape ■	Illegal Operation ⊠
70/242 70/243 Printers	Channel 12 Sensed ⧄	Channel 9 Sensed ⧄	Not Used	Print Line Incomplete ■	Non-representative Code ⧄	Hold (See Note 2) ■	End-of-Forms ⧄	Illegal Operation ⊠
70/248 Bill Feed Printer	Channel 12 Sensed ⧄	Channel 9 Sensed ⧄	Print Check ■	Trans-mission Parity Error ■	Code Error ⧄	Hold (See Note 2) ■	End-of-Forms ⧄	Illegal Operation ⊠
70/432 Magnetic Tape Unit 70/442 Magnetic Tape Unit 70/445 Magnetic Tape Station	Read Error or Read After Write Error ■	Service Request Not Honored ■	Data block Greater than Count ⧄	Trans-mission Error ■	Magnetic Tape Alarm ■	BT/ET ⧄	Tape Mark ⧄	Illegal Operation (See Note 1) ⊠

■ = Unrecoverable Error ⧄ = Abnormal Condition at Termination ⊠ = Terminate the Program

Notes: 1. If the illegal operation occurred as a result of a Write Command, the operator is notified and can, via a reply, cause control to be returned to the program. In this case, the unrecoverable error bit in the CCB is set.

2. When a hold condition occurs, error recovery cycles on the I/O command until the operator removes the device from HOLD.

I/O Sense Byte Conditions (Direct Access Devices)

SEEK Operations (Direct-Access Devices)

Condition	Sense Byte #1	Sense Byte #2	Sense Byte #3	Meaning
Card Extract Counter Equals a Preset Limit.			2^0 (Card Extract Counter Equals Limit.)	Unrecoverable Error
An Attempt To Issue An Invalid Chain		2^2 Invalid Sequence		Unrecoverable Error
Processor Sends Less Than Six Bytes to Controller	2^0 Command Code Reject			Unrecoverable Error
File Mask Prohibits Seek	2^0 Command Code Reject	2^4 File Protected		Unrecoverable Error
Invalid Seek Address Or Controller Can Not Complete Seek	2^5 Seek Check			Unrecoverable Error
A Seek is Issued To a 70/568 and the Magazine Is Missing.	2^5 Seek Check		2^2 Missing Magazine	Unrecoverable Error

Search Operations (Direct-Access Devices)

Condition	Sense Byte #1	Sense Byte #2	Sense Byte #3	Meaning
2^3 Bit of Command Code = 0 and End of Track is Detected.		2^3 Not Found		Abnormal Condition at Termination.
An Attempt to Issue an Invalid Chain		2^2 Invalid Sequence		Unrecoverable Error
Incorrect Head # When Head Switching Occurs (2^3 bit of Command Code = 1)	2^0 Command Code Reject 2^2 Automatic Head Switching Error			Unrecoverable Error
End of Track is Detected and File Mask Prohibits Head Switching (2^3 bit of Command Code = 1)	2^0 Command Code Reject 2^2 Automatic Head Switching Error	2^4 File Protected		Unrecoverable Error
Transmission Parity Error While Transferring Data from Main Memory to the Controller.	2^4 Transmission Parity Error			Unrecoverable Error
Service Request Not Honored While Transferring Data from Main Memory to the Controller	2^6 Service Request Not Honored			Unrecoverable Error
Cyclic Check Bytes do not Verify in the key and/or data Field Read by the Controller.	2^1 Read Parity Error			Unrecoverable Error
Cyclic Check Bytes do not Verify in Count Field Read by the Controller	2^1 Read Parity Error	2^7 Count Field Data Error		Unrecoverable Error
Missing Address Markers or the Count Fields of Two Successive Records are Read and 2^7 Bit of Both Flag Bytes are Zero	2^1 Read Parity Error	2^5 Missing Address Markers		Unrecoverable Error
End of Cylinder is Detected		2^1 End of Cylinder 2^3 Not Found		Abnormal Condition at Termination

Search Operations (Direct–Access) (Cont'd)

Condition	Sense Byte #1	Sense Byte #2	Sense Byte #3	Meaning
Defective Track (no address or invalid address in defective or alternate track).	2^2 Automatic Head Switching Error 2^3 Track Check			Unrecoverable Error
Head Switching Error while on an alternate track.	2^2 Automatic Head Switching Error 2^3 Track Check			Unrecoverable Error
Command Code Reject	2^0 Command Code Reject			Unrecoverable Error
Unrecoverable error during error recovery*				Unrecoverable Error

*This condition can be caused by one of the following:

1. Error recovery was unable to read the Home Address or the Track Descriptor Record on an alternate or defective track.
2. Error recovery was unable to search or read data from an alternate track.

READ Operations (Direct-Access Devices)

Condition	Sense Byte #1	Sense Byte #2	Sense Byte #3	Meaning
2^3 Bit of Command Code = 1 and End of Cylinder is Detected		2^1 End of Cylinder 2^3 Not Found		Abnormal Condition at Termination
End of File Detected on any Read but a Read Home Address	2^1 End of File			Abnormal Condition at Termination
Incorrect Head # When Head Switching Occurs (2^3 bit of Command Code = 1)	2^0 Command Code Reject			Unrecoverable Error
End of Track Detected and File Mask Prohibits Head Switching (2^3 bit of Command Code=1)	2^0 Command Code Reject 2^2 Automatic Head Switching Error	2^4 File Protected		Unrecoverable Error
An Attempt to Issue an Invalid Chain		2^2 Invalid Sequence		Unrecoverable Error
2^3 Bit of Command Code = 0 and a Read Home Address or Read Track Descriptor Record is Issued and a Home Address or Track Descriptor Record is Missing		2^3 Not Found		Abnormal Condition at Termination
Service Request Not Honored While Transferring Data from the Controller to Main Memory	2^0 Service Request Not Honored			Unrecoverable Error
Cyclic Check Bytes do not Verify in the key and/or data Field Read by the Controller.	2^7 Read Parity Error			Unrecoverable Error
Missing Address Markers or the Count Fields of Two Successive Records are Read and 2^7 Bit of Both Flag Bytes are Zero	2^7 Read Parity Error	2^5 Missing Address Markers		Unrecoverable Error
Cyclic Check Bytes do not Verify in a Count Field Read by the Controller	2^7 Read Parity Error	2^7 Count Field Data Error		Unrecoverable Error

READ Operations (Direct-Access Devices) (Cont'd)

Condition	Sense Byte #1	Sense Byte #2	Sense Byte #3	Meaning
Command Code Reject	2^0 Command Code Reject			Unrecoverable Error
Defective Track (no address or invalid address in defective or alternate track)	2^2 Automatic Head Switching Error 2^3 Track Check			Unrecoverable Error
Head Switching Error while on an alternate track.	2^2 Automatic Head Switching Error 2^3 Track Check			Unrecoverable Error
Unrecoverable error during error recovery*				Unrecoverable Error

*This condition can be caused by one of the following:

1. Error recovery was unable to read the Home Address or the Track Descriptor Record on an alternate or defective track.
2. Error recovery was unable to search or read data from an alternate track.

All Operations

Condition	Sense Byte #1	Sense Byte #2	Sense Byte #3	Meaning
Device was inoperable at termination of the I/O operation.				Unrecoverable Error
Error recovery was entered and there was no seek in the chain.				Unrecoverable Error

WRITE Operations (Direct-Access Devices)

Condition	Sense Byte #1	Sense Byte #2	Sense Byte #3	Meaning
Write End of File With Any Write but a Write Track Descriptor Record or a Write Count, Key, Data Command	2^1 End of File			Abnormal Condition at Termination
Too many Bytes sent from Processor	2^0 Command Code Reject			Unrecoverable Error
File Mask Prohibits Execution of Specified Command	2^0 Command Code Reject	2^4 File Protected		Unrecoverable Error
A Write Track Descriptor Record or Write Count, Key, Data Command is Not Complete and Track End is Detected		2^0 Track End		Abnormal Condition at Termination
An Attempt to Issue an Invalid Chain		2^2 Invalid Sequence		Unrecoverable Error
Transmission Parity Error While Transferring Data from Main Memory to the Controller	2^4 Transmission Parity Error			Unrecoverable Error
Service Request Not Honored While Transferring Data from Main Memory to the Controller	2^0 Service Request Not Honored			Unrecoverable Error
Missing Address Markers on Count Field Which Must be Read in Order to Perform the Write, Write Key, Data and Write Data Commands		2^5 Missing Address Markers		Unrecoverable Error
Cyclic Check Bytes do not Verify on the Count Field Which Must be Read in Order to Perform the Write	2^7 Read Parity Error	2^7 Count Field Data Error		Unrecoverable Error
A Parity Error Occurs During a Write to the 70/568 Mass Storage Unit	2^7 Read After Write Parity Error			Unrecoverable Error

Executive Processing of I/O Operations

I/O Requests

♦ When an EXCP or EXCPW macro is executed, control is transferred to the Executive to initiate the I/O operation.

If a channel error condition (program check, protection check, or channel control check in channel status byte) occurs during I/O initiation, the I/O operation is not initiated and the following occurs:

1. The condition code is stored in the CCB.
2. The termination and unrecoverable error bits are set in the CCB.
3. The channel registers in scratch-pad memory, for the channel concerned are stored in bytes 16-31 of the CCB.
4. The accept unrecoverable error bit in the CCB is tested and:
   a. If it is set to 0 a message is typed to the operator indicating the error condition and the job is terminated.
   b. If it is set to 1 control is transferred to the program.

If a device inoperable condition occurs during I/O initiation, the I/O operation is not initiated and a message is typed to the operator. When the operator has made the device operable, he keys in the proper response and the I/O is reissued by the Executive. If the condition can not be corrected, the operator has the option of bypassing the error or giving control to the program (the accept unrecoverable error bit in the CCB is tested). If control is given to the program (the accept unrecoverable error bit is set to 1), the following information is stored in the CCB.

1. The condition code.
2. The termination and unrecoverable error bits are set.
3. The channel registers in scratch-pad memory for the channel concerned are stored in bytes 16-31 of the CCB.

If the program does not accept control (the accept unrecoverable error bit is set to 0), the job is terminated.

The program can determine if the I/O operation was initiated by testing the condition code in the CCB. A condition code other than $(00)_2$ indicates that the I/O operation was not initiated and that the information contained in the channel status byte and/or the standard device byte must be tested.

I/O Interrupts

♦ Executive processing of input/output interrupts that occur after an I/O operation has been initiated is at two levels as follows:

Successful Interrupt - A successful interrupt is one that occurs due to a program controlled interrupt or a termination interrupt and there are no channel errors, device errors, or abnormal conditions. Executive processing of successful interrupts is as follows:

1. If the interrupt is a program controlled interrupt:

   a. The PCI bit in the Executive flag byte of the CCB is set.

   b. The Channel Address Register is stored in bytes 36-39 of the CCB.

2. If the interrupt is a termination:

   a. The termination bit in the Executive flag byte of the CCB is set.

   b. The channel registers for the channel concerned are stored in bytes 16-31 of the CCB.

Exceptional Interrupt - An exceptional interrupt is one that indicates that an unusual condition has occurred or that the I/O operation was unsuccessful. The following types of exceptional interrupts can occur:

Channel Errors - If a channel error occurs during the execution of an I/O operation, the following occurs:

1. The termination and unrecoverable error bits are set in the Executive Flag Byte of the CCB.

2. The channel registers in scratch-pad memory for the channel concerned are stored in bytes 16-31 of the CCB.

3. The accept unrecoverable error bit in the User Flag Byte of the CCB is tested and:

   a. If it is set to 0 a message is typed to the operator indicating the error condition and the job is terminated.

   b. If it is set to 1 control is transferred to the program.

      Note: If a magnetic tape station becomes inoperable during the execution of an I/O operation, the procedure described under channel errors occurs.

Device Errors - If a device error occurs during the execution of an I/O operation, the Executive either performs or does not perform standard error recovery depending upon the setting of the avoid device error recovery bit in the user flag byte of the CCB (if 0, standard error recovery is performed; if 1, standard error recovery is not performed).

I/O Interrupts (Cont'd)

Standard error recovery performed - If standard error recovery is performed and is successful, processing continues and no error is indicated. If standard error recovery is performed and is unsuccessful:

1. A message is typed to the operator indicating the error.
2. If the operator uses a reply of 1, the accept unrecoverable error bit in the user flag byte of the CCB is tested and (a) if set to 0 the job is terminated or (b) if set to 1 control is transferred to the program.

Standard error recovery is not performed - If standard error recovery is not performed, the accept unrecoverable error bit in the user flag byte of the CCB is tested and (a) if set to 0 the job is terminated or (b) if set to 1 control is transferred to the program.

Device Errors - Direct Access Devices - Device errors occurring on direct-access devices are handled in the same way as other device errors with the following exception: if standard error recovery is performed and is unsuccessful, the avoid direct access error recovery messages bit in the user flag byte of the CCB is tested and:

1. If set to 0 a message is typed to the operator indicating the error.
2. If set to 1 error messages to the operator are bypassed and the job is terminated or control is transferred to the program depending on the setting of the accept unrecoverable error bit in the user flag byte of the CCB.

   Note: If an unrecoverable device error (all devices including direct-access) occurs and the program specifies that it wants control (accept unrecoverable error bit is set to 1), the following information is supplied in the CCB before control is transferred to the program.

   1. The termination and unrecoverable error bits are set in the Executive Flag Byte of the CCB.
   2. The channel registers in scratch-pad memory for the channel concerned are stored in bytes 16-31 of the CCB.
   3. The device sense bytes are stored in bytes 32-34 of the CCB.

Abnormal Conditions - If an abnormal condition (i.e., BT/ET sensed, tape mark sensed, etc.) is indicated when an I/O operation terminates, the following occurs before control is transferred to the program.

1. The termination and abnormal condition at termination bits in the Executive flag byte of the CCB are set.
2. The channel registers in scratch-pad memory for the channel concerned are stored in bytes 16-31 of the CCB.
3. The device sense bytes are stored in bytes 32-34 of the CCB.

EXCP Execute Channel Program

GENERAL DESCRIPTION

♦ The EXCP macro is issued to request that a physical I/O operation be performed. This macro is used in conjunction with the CCB macro and the CCW instruction. As soon as the I/O request has been logged by the I/O dispatch routine, the program is eligible for reentry.

FORMAT

♦ Format of the EXCP macro is as follows:

Name	Operation	Operand
	EXCP	Blockname

SPECIFICATION RULES

Name Field

♦ Not required.

Operation Field

♦ EXCP.

Operand Field

♦ The blockname entry must be the symbolic name (unique CCB name or DTF name-filename) associated with the CCB established for the particular device.

Physical FCP determines the device concerned from the Command Control Block specified by the blockname. The I/O request is initiated if the device is not busy or is placed in a queue for the appropriate channel if the device is busy. The program is now eligible for reentry.

The CHECK macro can be executed to determine the status of the I/O operation that was initiated by the EXCP.

EXCPW Execute Channel Program and Wait

GENERAL DESCRIPTION

♦ The EXCPW macro is issued to request that a physical I/O operation be performed and that the calling program be placed in a waiting condition until the I/O operation has been completed. This macro is used in conjunction with the CCB macro and the CCW instruction.

FORMAT

♦ The format of the EXCPW macro is as follows:

Name	Operation	Operand
	EXCPW	Blockname

SPECIFICATION RULES

Name Field

♦ Not required.

Operation Field

♦ EXCPW.

Operand Field

♦ The blockname entry specifies the symbolic name associated with the CCB (Command Control Block) established for the particular device.

Physical FCP determines the device concerned from the Command Control Block specified by the blockname. The I/O request is initiated if the device is not busy or is placed in a queue for the appropriate channel if the device is busy. The program is placed in a waiting state until the I/O has terminated. When the I/O has terminated, the program is eligible for reentry.

WAIT
Wait

GENERAL DESCRIPTION

♦ The WAIT macro is issued when the program requires that an I/O operation initiated by an EXCP macro be completed before execution of the program can continue.

FORMAT

♦ The format of the WAIT macro is as follows:

Name	Operation	Operand
	WAIT	Blockname

SPECIFICATION RULES

Name Field

♦ Not required.

Operation Field

♦ WAIT.

Operand Field

♦ The blockname entry specifies the symbolic name associated with the CCB (Command Control Block) established for the particular device.

When the I/O operation initiated by the EXCP macro has terminated, the program is eligible for reentry.

If a WAIT macro is executed without prior issuance of a corresponding EXCP macro, control is transferred to the program as if an I/O operation was completed.

CCW Define Channel Command Word

GENERAL DESCRIPTION

♦ The CCW instruction defines the information required to execute an I/O operation. The information specified in the CCW instruction generates an eight-byte Channel Command Word. CCW is a single instruction rather than a macro.

FORMAT

♦ The format of the CCW instruction is as follows:

Name	Operation	Operand
A symbol or not used	CCW	Command Code, Data Address, Flags, Byte Count

SPECIFICATION RULES

Name Field

♦ Any symbol or not required.

Operation Field

♦ CCW.

Operand Field

♦ All four operands must appear and they are written from left to right as follows:

1. The first operand (Command Code) is an absolute expression that specifies the command code which indicates the operation to be performed by the input/output device (see table 1-20).

Table 1-20. CCW Valid Command Codes

	Command Code								
Bit	2^7 0	2^6 1	2^5 2	2^4 3	2^3 4	2^2 5	2^1 6	2^0 7	Operation to be Performed
	M/0	M/0	M/0	M/0	0	0	0	1	Sense
	M/0	M/0	M/0	M/0	M/0	0	1	0	Read Reverse
	M/0	M/0	M/0	M/B/0	M/0	0	1	1	Write
	M/0	M/0	M/0	M/B/0	M/0	1	0	0	Write Erase
	M/0	M/0	M/0	M/B/0	M/0	1	0	1	Read
	M/0	M/0	M/0	M/0	0	1	1	1	Write Control
	M/0	M/0	M/0	M/0	1	0	0	1	Transfer in Channel

Operand Field
(Cont'd)

Notes:

a. The bit position designated as B indicates that the specified device is connected to the multiplexor channel and that the multiplexor is to be operated in the Burst Mode. If this position is a 1 bit, the multiplexor channel is "locked-on" to the selected device and the operation of other devices connected to the multiplexor channel is inhibited. If "B" is not used, this bit position must be 0.

b. Bit positions designated as M (modifier) indicate variations of the operation and are unique to the specific input/output device. If "M" is not required, these bit positions must be 0.

c. Command Code is an absolute expression which can be expressed decimally or hexadecimally. For example, 03 or X'03' both specify tape write. If command code is expressed decimally a leading zero can be omitted. For example, 05 and 5 both can be used to specify a tape read.

2. The second operand (Data Address) is an absolute or relocatable expression that specifes the memory address to or from which data is to be transferred or the memory address of the next CCW if the command is a Transfer in Channel, or the address of the control byte if this is a write control (command code 07). If read reverse, this area is the rightmost address of the read-in area.

3. The third operand (Flags) is an absolute expression that specifies the contents of the flag byte as follows:

Bit Position	Meaning
0	Chain Data (CD).
1	Chain Command (CC).
2	Suppress Length Indication (SLI).
3	Skip.
4	Program Controlled Interrupt (PCI).
5-7	Must be 0's.

The flag byte is normally expressed hexadecimally. For example, X'A0' can be used to direct that the SLI flag and the chain data flag be set. All these flags are concerned with data and command chaining. If chaining is not to be done, X'00' should be specified.

Bits 5-7 are not allowed to be set. Only bits 0-4 are allowed. The Assembler sets only the allowed bits requested. If unallowed bits are requested to be set the Assembler flags the CCW instruction with a warning (D) flag.

Operand Field *(Cont'd)*

4. The fourth operand (Byte Count) is an absolute expression that specifies the number of bytes to be transferred by this I/O operation. (The number of bytes specified can be from 1-65,536 or zero.)

   Note: If 65,536 bytes are specified, the Assembler translates this to a byte count of all zeros in the CCW. This is because the processor transfers the maximum number of bytes (65,536) when the CCW contains a byte count of all zeros. For any other allowable byte count from 1 to 65,535 the Assembler puts this amount in the byte count in the CCW and the processor transfers this byte count. If zero is specified the Assembler puts zero in the byte count in the CCW and the processor transfers 65,536 bytes when I/O operations occur.

Notes:

♦ 1. If there is a symbol in the Name field of the CCW instruction, it is assigned the address value of the leftmost byte of the channel command word. The length attribute of the symbol is eight.

2. Eight bytes are generated by the CCW instruction as shown in table 1-21.

Table 1-21. Eight-Byte CCW

Byte	Bit Position	Contents
0		Command Code. This is generated from the first operand of the CCW instruction.
1-3		Data or CCW Address. This is generated from the second operand of the CCW instruction.
4		Flags. This is generated from the third operand of the CCW instruction.
	0	Chain Data Flag (CD).
	1	Chain Command Flag (CC).
	2	Suppress Length Indicator Flag (SLI).
	3	Skip Flag.
	4	Program Controlled Interrupt (PCI).
	5-7	Must be 0's.
5		Set to 0's.
6-7		Byte Count. This is generated from the fourth operand of the CCW instruction.

Data Chaining

♦ In addition to transferring data to and from a single memory area, 70/35, 45, 55 programs can read into or write from many noncontiguous areas of memory by executing one EXCP macro. The EXCP macro points to a Command Control Block (CCB) which, in turn, points to a Channel Command Word (CCW). The program sets up a sequential string of CCW macros (chain) that are to be executed. Each CCW instruction designates an area of memory at which to continue the current operation. The chain data bit (bit 32) of each CCW (except the last) in the chain must be set to 1. When the byte count of one CCW has lapsed, the next CCW in sequence is automatically fetched and the current operation is continued at the memory area specified by the new CCW. This sequence continues until a CCW is fetched that has the chain data bit set to 0. After completion of the command specified by this CCW, the data chain is terminated.

Command Chaining

♦ Spectra 70/35, 45, 55 programs can specify several operations to an input/output device by executing one EXCP macro. The EXCP macro points to a Command Control Block (CCB) which, in turn, points to a Channel Command Word (CCW). The program sets up a sequential string of CCW instructions (chain) that are to be executed. Each CCW instruction specifies a single input/output operation to be performed. The chain command bit (bit 33) of each CCW (except the last) in the chain must be set to 1. When the operation specified by one CCW is completed, the next CCW in sequence is automatically fetched and its operation is initiated. This sequence continues until a CCW is fetched that has the chain command bit reset to 0. After completion of the command specified by this CCW, the command chain is terminated.

Suppress Length Indication

♦ Incorrect length occurs in the 70/35, 45, 55 Processors when the number of bytes specified in the Channel Command Word is not equal to the number of bytes sought by, or sent from the input/output device. (When a command or chain of commands terminates, the data byte count has not lapsed). If the suppress length indication bit (bit 34) in the CCW is set to 1, the program does not receive an indication of an incorrect length upon termination of the input/output operation. If the suppress length indication is set to 0, the program receives an indication of an incorrect length upon termination of the input/output operation. This indication is contained in the Command Control Block (CCB). If the SLI bit is set to 1 and the chain data bit is set to 0 the incorrect length indication is suppressed, if it occurs.

Skipping Data

♦ Spectra 70/35, 45, 55 programs can skip portions of a block of information during an input operation when data chaining. If the skip bit (bit 35) in a CCW is set to 1, the transfer of data to main memory specified by this CCW is suppressed. The skip bit can be used only with Read, Read Reverse, and Sense commands.

Program Controlled Interrupt

♦ During data and command chaining 70/35, 45, 55 programs can cause an input/output channel interrupt to occur when a Channel Command Word is fetched. If the program controlled interrupt bit (bit 36) in the CCW is set to 1, a channel interrupt occurs when this CCW is fetched from main memory and after the first data byte has been transferred. When the channel interrupt occurs, the Executive sets a bit in the Command Control Block (CCB) which can be interrogated by the program.

CHECK
Check

GENERAL DESCRIPTION

♦ The CHECK macro is a specialized version of the WAIT macro. It permits the program to check the status of an I/O operation previously initiated by an EXCP.

FORMAT

♦ The format of the CHECK macro is as follows:

Name	Operation	Operand
	CHECK	Blockname, branch address

SPECIFICATION RULES

Name Field

♦ Not required.

Operation Field

♦ CHECK.

Operand Field

♦ The blockname entry specifies the symbolic name associated with the CCB (Command Control Block) established for the particular device. The branch address entry is the symbolic name of the location to which a transfer is made if the I/O operation initiated by the EXCP macro is not completed. If the I/O operation is complete, the instruction following the CHECK macro is executed.

CPCI Check for PCI

GENERAL DESCRIPTION

♦ The CPCI macro determines if a Program Controlled Interrupt has occurred during an input/output operation.

FORMAT

♦ The format of the CPCI macro is as follows:

Name	Operation	Operand
	CPCI	Blockname, branch address

SPECIFICATION RULES

Name Field

♦ Not required.

Operation Field

♦ CPCI.

Operand Field

♦ The blockname entry specifies the symbolic name associated with the CCB (Command Control Block) established for the particular device. The branch address entry is the symbolic name of the location to which a transfer is made if the CCB indicates that a PCI has *not* occurred. If a PCI has occurred, the instruction following the CPCI macro is executed.

FCP MACRO EXPANSION

File Descriptor Macros

DTFPH

Name	Symbolic Name	Byte	Bit Position	Contents
Filename	CCB	0-39		Channel Command Block for reading or writing.
	CCW1	40-47		Channel Command Word for reading or writing for this tape file.
	CCW2	48-55		Channel Command Word for Write Control commands. Used during error recovery at which time CCW1 and CCW2 are chained together. Used for other purposes when DTFPH is an inner macro.
	BYTCNT	56-57		Contains the total number of bytes (binary) contained in the DTF (DTFPH, DTFSR, DTFDA) macro.
	FILABL	58	0-1	Contains indication of type of label: Standard = $(00)_2$ Nonstandard = $(01)_2$ No Label = $(11)_2$
	TYPEFLE	58	2-5	Contains indication of type of file: Input = $(0000)_2$ Output = $(0001)_2$
	FILSTAT	58	6-7	Indicates the status of a file at particular point in time. The OPEN and CLOSE routines modify this field.
	ERRORB	59		Used by error recovery routines and must not be modified by the program. Error recovery initializes and modifies this field.
	MRKCTR	60-61		Contains the number of tape marks (binary) to be passed when positioning tape at OPEN time.

FCP MACRO EXPANSION

File Descriptor Macros

DTFPH *(Cont'd)*

Name	Symbolic Name	Byte	Bit Position	Contents
Filename (Cont'd)	DTFTYPE	62		Indicates the type of file definition: DTFPH = P DTFSR = S DTFDA = D
	FIRSTIM	63	0	OPEN = 1 $(2)^7$ CLOSEd = 0 $(2)^7$
	RESERV	63	1-5	Reserved for FCP usage.
	REWIND	63	6-7	Indicates type of rewind: Rewind = $(11)_2$ Rewind and unload = $(01)_2$ No rewind = $(00)_2$
	LABADDR	64-67		Address of user routine to check or build additional labels.
	LABNAME	68-71		*Tape files* - address of where the file's standard label information is located. *Direct-access files* - Address where FCP builds the file's extent matrix.
	IOAREA1	72-75		Address of where labels are read and written or IOAREA1.
	FILENAME	76-82		Contains the name field specified in the DTF (DTFPH, DTFSR, DTFDA).
	ISRKEY	83		Contains the number of bytes (binary) from 1-256 in the key field if keys are to be read or written. If KEYLEN (DTFDA) or ISRKEY (DTFSR) is not specified, this byte is all 0's.
	IDLER	84-87		Reserved for FCP usage.

FCP MACRO EXPANSION

File Descriptor Macros

DTFPH *(Cont'd)*

Name	Symbolic Name	Byte	Bit Position	Contents
Filename (Cont'd)	OVRTN	88-91		Address of user routine to which FCP transfers following completion of its end-of-volume processing.
	FILENAMB	92-95		This name is generated and is always composed of the name field specified in the DTFPH plus the character "B". Prior to closing the file, the program must move the block count to be included in the trailer label to this name.
	EXPAND	96-97		Contains the primary and alternate device IF ALTDEV=NAME has been specified.
	ALTTAPE	98-103		Contains the symbolic device name (nnnnnn) for a tape device that is used as an alternate when a tape file has two or more reels (volumes) of data.
	WHATSIT	104-111		Reserved for future use.
	LABELRW	112-191		Generated only for physical level processing (DTFPH) of tape files and is the area where standard labels are read and written (80 bytes).
		112-263		Generated only for physical level processing (DTFPH) of direct-access files and is the area where standard labels are read and written (152 bytes).

FCP MACRO EXPANSION

File Descriptor Macros

DTFSR-Magnetic Tape and Direct-Access Devices

Name	Symbolic Name	Byte	Bit Position	Contents
Filename	DTFPH	0-111		Contains contents of DTFPH expansion (bytes 0-111).
	EOFADDR	112-115		Address of user end-of-file routine.
	IOAREA2	116-119		Address of IOAREA2.
	BLKSIZE	120-123		Number of bytes of memory(binary) in IOAREA1 and IOAREA2.
	RECSIZE	124-127		Number of bytes to be transferred to or from memory for fixed-length blocked records. Address where general register is stored that contains the length of each record for undefined records.
	LHECON	128-131		Indicates the next record available for deblocking or the next area available for record movement. This entry is updated on each GET or PUT of a record.
	AFINAL	132-135		Contains the address of the last data byte read or written +1 stored after each I/O operation.
	RECFORM	136	0-2	Contains indication of record format: FIXUNB = $(000)_2$ FIXBLK = $(001)_2$ VARUNB = $(010)_2$ VARBLK = $(011)_2$ UNDEF = $(100)_2$
	WORKA	136	3	Indicates if a work area is specified: YES = $(1)_2$ NO = $(0)_2$

FCP MACRO EXPANSION

File Descriptor Macros

DTFSR-Magnetic Tape and Direct-Access Devices *(Cont'd)*

Name	Symbolic Name	Byte	Bit Position	Contents
Filename (Cont'd)	READ	136	4	For tape files, indicates direction in which the tape is to be read. FORWARD $=(0)_2$ BACK $=(1)_2$
	TPMARK	136	5	Indicates absence of a tape mark before the first data block: Tape mark absent $=(0)_2$ Tape mark not absent $=(1)_2$
	OVERFLO	136	6	Indicates if record overflow feature is to be used (direct-access files): YES $=(1)_2$ NO $=(0)_2$
	CKPTREC	136	7	Indicates if a tape input file contains checkpoint records interspersed among data records: YES $=(1)_2$ NO $=(0)_2$
	TESTSW0	137	0	Generated indicators used by the system.
	TESTSW1	137	1	
	TESTSW2	137	2	
	TESTSW3	137	3	
	TESTSW4	137	4	
	TESTSW5	137	5	
	TESTSW6	137	6	
	TESTSW7	137	7	
	B1STAT	138		Contains the status of IOAREA1
	B2STAT	139		Contains the status of IOAREA2

FCP MACRO EXPANSION

File Descriptor Macros

DTFSR-Magnetic Tape and Direct-Access Devices *(Cont'd)*

Name	Symbolic Name	Byte	Bit Position	Contents
Filename (Cont'd)	FPSWRD	140-143		Generated internal switches.
	FPSWWR	144-147		
	FPSWA	148-151		
	FPSWB	152-155		
	FPSWC	156-159		
	FPSWD	160-163		
	FPSWLB	164-167		
	FPSWTR	168-171		
	IOREG	172-175		Points to the storage area for the general register that contains the absolute base address of the current record each time a GET or PUT is issued.
	BUFFER	176-179		Reserved for future expansion.
	BUFFER1	180-190		Reserved for future expansion.
	CONTROLP	191		Storage area for control byte if a CNTRL macro is issued to the file.
	ERROPT	192-195		Contains the address of a user routine to transfer to if an error occurs on a tape input file (if Name specified in DTFSR entry ERROPT).
	VARBLD	196-199		Reserved for FCP usage.
	WLRERR	200-203		Address of user routine if a wrong length record is read.
	ISRSEEK	204-207		Address of track reference field.

Notes: 1. For direct-access files, up to 29 additional CCW's and a 56-byte work area are appended to the DTFSR. These additions begin with relative byte number 208.

2. The following fields must be defined by the user and appear in the DTFPH expansion (bytes 0-111):

FILABLE DEVICE IOAREA1
DEVADDR (Tape only)

FCP MACRO EXPANSION

File Descriptor Macros

DTFSR-Card Readers (70/237)

Name	Symbolic Name	Byte	Bit Position	Contents
Filename	DTFPH	0-111		Contains contents of DTFPH expansion-bytes 0-111.
	EOFADDR	112-115		Address of user end-of-file routine-/* recognized in first two columns of card read.
	IOAREA2	116-119		(See DTFSR-Magnetic Tape.)
	BLKSIZE	120-123		Number of bytes of memory (binary) in IOAREA1 and IOAREA2. This field can not exceed 80 bytes.
	RECSIZE	124-127		Not used.
	LHECON	128-131		(See DTFSR - Magnetic Tape)
	AFINAL	132-135		(See DTFSR-Magnetic Tape.)
	RECFORM	136	0-2	Contains the type of record format and must be FIXUNB=$(000)_2$.
	WORKA	136	3	(See DTFSR-Magnetic Tape.)
	UNUSED1	136	4-7	Not used.
	TESTSW0	137	0	Generated Indicators used by the system.
	TESTSW1	137	1	
	TESTSW2	137	2	
	TESTSW3	137	3	
	TESTSW4	137	4	
	TESTSW5	137	5	
	TESTSW6	137	6	
	TESTSW7	137	7	
	B1STAT	138		(See DTFSR - Magnetic Tape)
	B2STAT	139		(See DTFSR - Magnetic Tape)

FCP MACRO EXPANSION

File Descriptor Macros

DTFSR-Card Readers (70/237,70/251-Card Read Only) *(Cont'd)*

Name	Symbolic Name	Byte	Bit Position	Contents
Filename (Cont'd)	FPSWRD	140-143		Generated internal switches.
	FPSWWR	144-147		
	FPSWA	148-151		
	FPSWB	152-155		
	FPSWC	156-159		
	FPSWD	160-163		
	FPSWLB	164-167		
	FPSWTR	168-171		
	IOREG	172-175		(See DTFSR Magnetic Tape.)
	BUFFER	176-179		Reserved for future expansion.
	BUFFER1	180-191		Reserved for future expansion.

Note: The following fields must be defined by the user and appear in the DTFPH expansion (bytes 0-111):

DEVICE

IOAREA1

DEVADDR

FCP MACRO EXPANSION

File Descriptor Macros

DTFSR-Printers (70/242,243,248)

Name	Symbolic Name	Byte	Bit Position	Contents
Filename	DTFPH	0-111		Contains contents of DTFPH expansion bytes 0-111.
	FILL	112-113		Not used.
	PRINTOV	114		Indicates if the program is to issue PRTOV macros. If nonzeros, PRTOV macros are to be issued. If 0's, PRTOV macros are not to be issued.
	SSAFTER	115		Reserved for FCP usage.
	IOAREA2.	116-119		Address of IOAREA2.
	BLKSIZE	120-123		Number of bytes of memory (binary) in IOAREA1 and IOAREA2
	RECSIZE	124-127		Number of a general register that contains the length of each record for undefined records.
	LHECON	128-131		(See DTFSR - Magnetic Tape)
	AFINAL	132-135		(See DTFSR-Magnetic Tape.)
	RECFORM	136	0-2	Contains indication of record format: FIXUNB = $(000)_2$ VARUNB = $(010)_2$ UNDEF = $(100)_2$
	WORKA	136	3	(See DTFSR - Magnetic Tape.)

FCP MACRO EXPANSION

File Descriptor Macros

DTFSR-Printers (70/242,243,248) *(Cont'd)*

Name	Symbolic Name	Byte	Bit Position	Contents
Filename (Cont'd)	CTLCHR	136	4	Indicates if a control character is part of each record: If $(1)_2$ - control character is part of each record. If $(0)_2$ - control character is not part of each record.
	UNUSED1	136	5-7	Not used.
	TESTSW0	137	0	Generated indicators used by the system.
	TESTSW1	137	1	
	TESTSW2	137	2	
	TESTSW3	137	3	
	TESTSW4	137	4	
	TESTSW5	137	5	
	TESTSW6	137	6	
	TESTSW7	137	7	
	B1STAT	138		(See DTFSR - Magnetic Tape)
	B2STAT	139		(See DTFSR - Magnetic Tape)
	FPSWRD	140-143		Generated internal switches.
	FPSWWR	144-147		
	FPSWA	148-151		
	FPSWB	152-155		
	FPSWC	156-159		
	FPSWD	160-163		
	FPSWLB	164-167		
	FPSWTR	168-171		
	IOREG	172-175		(See DTFSR - Magnetic Tape.)

FCP MACROS EXPANSION

File Descriptor Macros

DTFSR-Printers (70/242,243,248) *(Cont'd)*

Name	Symbolic Name	Byte	Bit Position	Contents
Filename (Cont'd)	BUFFER	176-179		Reserved for future expansion.
	BUFFER1	180-190		Reserved for future expansion.
	CONTROLP	191		(See DTFSR-Magnetic Tape.)

Note: The following fields must be defined by the user and appear in the DTFPH expansion (bytes 0-111):

DEVADDR

TYPEFLE

DEVICE

IOAREA1

FCP MACRO EXPANSION

File Descriptor Macros

DTFSR-Paper Tape Reader/Punch (70/221)

Name	Symbolic Name	Byte	Bit Position	Contents
Filename	DTFPH	0-111		Contains contents of DTFPH expansion - bytes 0-111.
	EOFADDR	112-115		Address of user end-of-file routine /* recognized in first two bytes of record read.
	IOAREA2	116-119		Address of IOAREA2.
	BLKSIZE	120-123		Number of bytes of memory (binary) in IOAREA1 and IOAREA2.
	RECSIZE	124-127		Number of a general register that contains the length of each record for undefined records.
	LHECON	128-131		(See DTFSR - Magnetic Tape)
	AFINAL	132-135		(See DTFSR - Magnetic Tape.)
	RECFORM	136	0-2	Contains indication of record format: FIXUNB = $(000)_2$ UNDEF = $(100)_2$
	WORKA	136	3	(See DTFSR - Magnetic Tape.)
	READ	136	4	Indicates the direction in which the paper tape is to be read: FORWARD = $(0)_2$ BACK = $(1)_2$
	UNUSED	136	5-7	Not used.

FCP MACRO EXPANSION

File DescriptorMacros

DTFSR-Paper Tape Reader/Punch (70/221) *(Cont'd)*

Name	Symbolic Name	Byte	Bit Position	Contents
Filename (Cont'd)	TESTSW0	137	0	Generated indicators used by the system.
	TESTSW1	137	1	
	TESTSW2	137	2	
	TESTSW3	137	3	
	TESTSW4	137	4	
	TESTSW5	137	5	
	TESTSW6	137	6	
	TESTSW7	137	7	
	B1STAT	138		(See DTFSR - Magnetic Tape)
	B2STAT	139		(See DTFSR - Magnetic Tape)
	FPSWRD	140-143		Generated internal switches.
	FPSWWR	144-147		
	FPSWA	148-151		
	FPSWB	152-155		
	FPSWC	156-159		
	FPSWD	160-163		
	FPSWLB	164-167		
	FPSWTR	168-171		
	IOAREG	172-175		(See DTFSR - Magnetic Tape.)
	TRANS	176-179		Address of translation table if required.
	BUFFER1	180-191		Reserved for future expansion.

Note: The following fields must be defined by the user and appear in the DTFPH expansion (bytes 0-111):

DEVADDR	DEVICE
TYPEFLE	IOAREA1

FCP MACRO EXPANSION

File Descriptor Macros

DTFSR - Card Punches (70/234, 236)

Name	Symbolic Name	Byte	Bit Position	Contents
Filename	DTFPH	0-111		Contains contents of DTFPH expansion bytes 0-111.
	CRDERR	112-115		Contains the address of an area used as a backup for repunching cards (see CARDS).
	IOAREA2	116-119		Address of IOAREA2.
	BLKSIZE	120-123		Number of bytes of memory (binary) in IOAREA1 and IOAREA2.
	RECSIZE	124-127		Number of a general register that contains the length of each record for undefined records.
	LHECON	128-131		(See DTFSR - Magnetic Tape)
	AFINAL	132-135		Contains the address of the last data byte punched +1 stored after each I/O operation.
	RECFORM	136	0-2	Contains indication of record format: FIXUNB = $(000)_2$ VARUNB = $(010)_2$ UNDEF = $(100)_2$
	WORKA	136	3	Indicates if work area is specified: YES = $(1)_2$ NO = $(0)_2$
	UNUSED1	136	4-7	Reserved for FCP usage.

FCP MACRO EXPANSION

File Descriptor Macros

DTFSR - Card Punches (70/234, 236) *(Cont'd)*

Name	Symbolic Name	Byte	Bit Position	Contents
Filename (Cont'd)	TESTSW0	137	0	Generated indicators used by the system.
	TESTSW1	137	1	
	TESTSW2	137	2	
	TESTSW3	137	3	
	TESTSW4	137	4	
	TESTSW5	137	5	
	TESTSW6	137	6	
	TESTSW7	137	7	
	B1STAT	138		(See DTFSR - Magnetic Tape)
	B2STAT	139		(See DTFSR - Magnetic Tape)
	FPSWRD	140-143		Generated internal switches.
	FPSWWR	144-147		
	FPSWA	148-151		
	FPSWB	152-155		
	FPSWC	156-159		
	FPSWD	160-163		
	FPSWLB	164-167		
	FPSWTR	168-171		
	IOREG	172-175		Points to the storage area for the general register that contains the absolute base address of the current record each time a PUT is issued.
	BUFFER	176-179		Reserved for future expansion.
	BUFFER1	180-191		Reserved for future expansion.
	CARDS	192-351		Area used as backup if CRDERR = RETRY is specified and a punch error occurs.

Note: The following fields must be defined by the user and appear in the DTFPH expansion (bytes 0-111):

DEVADDR DEVICE IOAREA1

FCP MACRO EXPANSION

File Descriptor Macros

DTFDA-Direct-Access Devices

Name	Symbolic Name	Byte	Bit Position	Contents
Filename	DTFPH	0-111		Contains contents of DTFPH expansion - bytes 0-111.
	AFTER	112-115		If AFTER not specified = 0's. If AFTER specified = address of the CCW's to handle this option.
	AFTERID	116-119		If AFTERID not specified = 0's. If AFTERID specified = address of the CCW's for the option.
	BLKSIZE	120-123		Contains the maximum number of bytes (binary) that can be transferred to or from IOAREA1 at any one time.
	RECSIZE	124-127		Contains (in the least significant four bits) the number of the general register that contains the record length if undefined records are specified. If RECSIZE not specified, this field is all 0's.
	WRITEID	128-131		If WRITEID not specified = 0's. If WRITEID specified = address of the CCW's to handle this option.
	WRITEKY	132-135		If WRITEKY not specified = 0's. If WRITEKY specified = address of the CCW's to handle this option.
	CONTROL	136-139		If CONTROL not specified = 0's. If CONTROL specified = address of the CCW's to handle this option.
	RECFORM	140	0-2	Contains indication of record format: FIXUNB = $(000)_2$ UNDEF = $(100)_2$

FCP MACRO EXPANSION

File Descriptor Macros

DTFDA-Direct-Access Devices *(Cont'd)*

Name	Symbolic Name	Byte	Bit Position	Contents
Filename (Cont'd)	SRCHM	140	3	Indicates if extended searching is to be done: YES = $(1)_2$ NO = $(0)_2$
	CAPREC	140	4	Indicates if capacity record option is desired: YES = $(1)_2$ NO = $(0)_2$
	RELADDR	140	5	Indicates if the track address in the track reference field is a relative address: Relative = $(1)_2$. Actual = $(0)_2$.
	UNUSED9	140	6-7	Reserved for future use.
	KEYLEN	141		If keys are to be referred to, contains the number of bytes (binary) in the key field of the records. If keys are not to be referred to, this field is 0's.
	DARESER	144-147		Reserved for future use.
	READID	148-151		If READID not specified = 0's. If READID specified = address of the CCW's to handle this option.
	READKY	152-155		If READKY not specified = 0's. If READKY specified = address of the CCW's to handle this option.
	KEYARG	156-159		Contains the address of an area in the user program in which the key is supplied before a Read or Write Key operation is executed. If the option is not specified, this field is 0's.

FCP MACRO EXPANSION

File Descriptor Macros

DTFDA-Direct-Access Devices *(Cont'd)*

Name	Symbolic Name	Byte	Bit Position	Contents
Filename (Cont'd)	IDLOC	160-163		Contains the address of an area in the user program that receives the appropriate ID (CCHHR) after each Read or Write. If the option is not specified, this field is 0's.
	SEEKADR	164-167		Contains the address of the track reference field in the user program.
	ERRBYTE	168-171		Contains the address of the ERROR/STATUS byte in the user program where FCP stores conditions when a WAITF is issued.
	FPIND1 FPIND2 FPIND3	172 173 174		Reserved for FCP usage.
	IAFTIND	175		$(80)_{16}$ indicates either AFTER or AFTERID=YES $(00)_{16}$ indicates neither AFTER nor AFTERID=YES
	LSTBLK	176-183		Contains the track address (MBBCCHHR) of the last record referenced.
	LSTNTRY	184-187		Contains the address of the entry in the matrix currently being used.
	LSTRLTR	188-191		Contains the highest relative track address in the extent previous to the current extent being processed.
	CCW's	192 +		Variable number of CCW's. From 4 to 46.

FCP MACRO EXPANSION

File Descriptor Macros

DTFEN

Name	Operation	Operand	Remarks
	DS	0F	
	DC	X'FFFFFFFFFFFFFFFF'	Indicates that all file expansions have been concluded.

At the time of assembly of the DTF macro-instructions all the options for file usage are determined. The DTFEN macro must check the global set switches and generate the appropriate FCP coding to service those processing options indicated as present in the preceding set of DTF's.

The DTFEN macro causes the coding necessary to service the processing options required by the preceding DTF's to be generated into the user program. The memory requirement for this coding is variable, dependent on the file description, usually averaging approximately 1,500 bytes.

FCP MACRO EXPANSION

OPEN

Name	Operation	Operand	Remarks
	CNOP	0,4	Needed only when location counter is not on a word boundary.
Name	LM	14,15,*+8	
	BAL	1,N(15)	N = a relative jump table location for this routine.
	DC	AL4 (*+8+4F)	F = number of files to be opened. This provides the return to the user's coding.
	DC	AL4 (IFCP)	Contains the jump address to FCP.
	DC	AL4 (Filename 1)	Contains the address of the DTF associated with this file.
		.	
		.	
		.	
	DC	AL4 (Filename N)	Contains the address of the DTF associated with the last file to be opened.
	EXTRN	IFCP	
	EXTRN	Filename 1	External only if DTF's are not a part of this module.
	EXTRN	Filename N	

LBRET

Name	Operation	Operand	Remarks
	CNOP	0,2	Needed only if location counter is not on a word boundary.
	LA	1, 1 or 2	Load register one with a 1 or 2 as specified in macro.
	BR	14	

FCP MACRO EXPANSION

CLOSE

Name	Operation	Operand	Remarks
	CNOP	0,4	Needed only if location counter is not on a word boundary.
Name	LM	14,15,*+8	
	BAL	1,N(15)	N = relative jump table location for this routine.
	DC	AL4 (*+8+4F)	F = number of files to be opened. This provides the return to the user's coding.
	DC	AL4 (IFCP)	Contains the jump address to FCP.
	DC	AL4 (Filename 1) . . .	Contains the address of the DTF associated with this file.
	DC	AL4 (Filename N)	Contains the address of the DTF associated with the last file to be opened.
	EXTRN	IFCP	External only if DTF's are not a part of this module.
	EXTRN	Filename 1	
	EXTRN	Filename N	

GET

Name	Operation	Operand	Remarks
	CNOP	0,4	Needed only when location counter is not on a word boundary.
Name	LM	14,1,*+8	
	B	N(15)	N = relative jump table location for this routine.
	DC	AL4 (*+16)	Provides the return to user coding.
	DC	AL4 (IFCP)	Contains the jump address to FCP.
	DC	AL4 (WORKA) or (0000) if no WORKA	Contains address of WORKA associated with this file.
	DC	AL4 (Filename)	Contains the address of the DTF for this file.
	EXTRN	IFCP	External only if DTF's are not part of this module.
	EXTRN	Filename	

FCP MACRO EXPANSION

PUT

Name	Operation	Operand	Remarks
	CNOP	0,4	Needed only when location counter is not on a word boundary.
Name	LM	14,1,*+8	
	B	N(15)	N = relative jump table location for this routine.
	DC	AL4 (*+16)	Provides the return to user coding.
	DC	AL4 (IFCP)	Contains the jump address to FCP.
	DC	AL4 (WORKA) or (0000) if no WORKA	Contains address of WORKA associated with this file.
	DC	AL4 (Filename)	Contains the address of the DTF for this file.
	EXTRN	IFCP	External only when DTF's are not part of this module.
	EXTRN	Filename	

RELSE

Name	Operation	Operand	Remarks
	CNOP	0,4	Needed only when location counter is not on a word boundary.
Name	LM	14,1,*+8	
	B	N(15)	N = relative jump table location for this routine.
	DC	AL4 (*+16)	Provides the return to user coding.
	DC	AL4 (IFCP)	Contains the jump address to FCP.
	DC	AL4 (0000)	
	DC	AL4 (Filename)	Contains the address of the DTF for this file.
	EXTRN	IFCP	External only if DTF's are not part of this module.
	EXTRN	Filename	

FCP MACRO EXPANSION

TRUNC

Name	Operation	Operand	Remarks
	CNOP	0,4	Needed only when location counter is not on a word boundary.
Name	LM	14,1,*+8	
	B	N(15)	N = relative jump table location for this routine.
	DC	AL4 (*+16)	Provides the return to user coding.
	DC	AL4 (IFCP)	Contains the jump address to FCP.
	DC	AL4 (0000)	
	DC	AL4 (Filename)	Contains the address of the DTF for this file.
	EXTRN	IFCP	External only if DTF's are not part of this module.
	EXTRN	Filename	

CNTRL

Devices other than Direct Access

Name	Operation	Operand	Remarks
	CNOP	0,4	Needed only if location counter is not on a word boundary.
Name	LM	14,1,*+8	
	B	N(15)	N = relative jump table location for this routine.
	DC	AL4 (*+16)	Provides the return to user coding.
	DC	AL4 (IFCP)	Contains the jump address to FCP.
	DC	X'000000BB'	BB = write control byte constructed from code, m, n combination.
	DC	AL4 (Filename)	Contains the address of the DTF for this file.
	EXTRN	IFCP	External only if DTF's are not a part of this module.
	EXTRN	Filename	

FCP MACRO EXPANSION

CNTRL

Direct-Access Devices (Non-Seek Operations)

Name	Operation	Operand	Remarks
	CNOP	0,4	Needed only if location counter is not on a word boundary.
Name	LM	14,1,*+8	
	B	N(15)	N = relative jump table location for this routine.
	DC	AL4 (*+16)	Provides the return to user coding.
	DC	AL4 (IFCP)	Contains the jump address to FCP.
	DC	AL4 (0000)	
	DC	AL4 (Filename)	Contains the address of the DTF for this file.
	EXTRN	IFCP	External only if DTF's are not a part of this module.
	EXTRN	Filename	

PRTOV

Name	Operation	Operand	Remarks
	CNOP	0,4	Needed only if location counter is not on a word boundary.
Name	LM	14,1,*+8	
	B	N(15)	N = relative jump table location for this routine.
	DC	AL4 (*+16)	Provides return to user coding.
	DC	AL4 (IFCP)	Contains the jump address to FCP.
	DC	C'N'	Specifies the tape channel (either 9 or 12) indicated in the second operand.
	DC	AL3 (Routine name) or C'000'	Contains the routine address if specified. in the third operand.
	DC	AL4 (Filename)	Contains the address of the DTF associated with this file.
	EXTRN	IFCP	External only if DTF's are not a part of this module.
	EXTRN	Filename	

FCP MACRO EXPANSION

FEOV

Name	Operation	Operand	Remarks
	CNOP	0,4	Needed only if location counter is not on a word boundary.
Name	LM	14,15,*+8	
	BAL	1, N(15)	N = relative jump table location for this routine.
	DC	AL4 (*+8+4F)	F = number of files to be forced. This provides the return to the user coding.
	DC	AL4 (IFCP)	Contains the jump address to FCP.
	DC	AL4 (Filename 1) . . .	Contains the address of the DTF associated with this file.
	DC	AL4 (Filename N)	Contains the address of the DTF associated with the last file to be forced.
	EXTRN	IFCP	External only if DTF's are not a part of this module.
	EXTRN	Filename 1	
	EXTRN	Filename N	

READ

Name	Operation	Operand	Remarks
	CNOP	0,4	Needed if location counter is not on a halfword boundary.
	LM	14,1,*+8	
	BR	N (0,15)	N = 64 for READID, N = 68 for READKY
	DC	AL4 (*+16)	Provides return to user coding.
	DC	AL4 (IDAMFCP) or (IRDID)	Contains the address for the FCP routine associated with the key or ID option.
	DC	AL4 (0000)	
	DC	AL4 (Filename)	Contains the address of the DTF associated with this file.
	EXTRN	IDAMFCP	External only if DTF's are not a part of this module.
	EXTRN	Filename	

FCP MACRO EXPANSION

WRITE

Name	Operation	Operand	Remarks
	CNOP	0,4	Needed only if location counter is not on a halfword boundary.
	LM	14,1,*+8	
	BR	N (0,15)	N = 72 for ID N = 76 for KEY N = 80 for AFTER N = 84 for AFTERID
	DC	AL4 (*+16)	Provides return to user coding.
	DC	AL4 (IDAMFCP)	
	DC	AL4 (0000)	
	DC	AL4 (Filename)	Contains the address of the DTF associated with this file.
	EXTRN	IDAMFCP	External only if DTF's are not a part of this module.
	EXTRN	Filename	

WAITF

Name	Operation	Operand	Remarks
	CNOP	0,4	Needed only if location counter is not on a halfword boundary.
	LM	14,1,*+8	
	BR	88 (0,15)	
	DC	AL4 (*+16)	Provides return to user coding.
	DC	AL4 (IDAMFCP)	Contains the address of the FCP routine for this macro.
	DC	AL4 (0000)	
	DC	AL4 (Filename)	Contains the address of the DTF associated with this file.
	EXTRN	IDAMFCP	External only if DTF's are not part of this module.
	EXTRN	Filename	

FCP MACRO EXPANSION

CNTRL

Direct-Access Devices (Seek Operations)

Name	Operation	Operand	Remarks
	CNOP	0,4	Needed only if location counter is not on a halfword boundary.
	LM	14,1,*+8	
	BR	92 (0,15)	
	DC	AL4 (*+16)	Provides return to user coding.
	DC	AL4 (IDAMFCP)	Contains the address of the FCP routine for this macro.
	DC	AL4 (0000)	
	DC	AL4 (Filename)	Contains the address of the DTF associated with this file.
	EXTRN	IDAMFCP	External only if DTF's are not a part of this module.
	EXTRN	Filename	

CCB

Name	Operation	Operand	Remarks
	CNOP	0,4	Needed only if location counter is not on a word boundary.
Name	DC	CL6 'sd'	sd = name of symbolic device to execute the CCB.
	DC	AL1 (dtc)	dtc = device class byte if specified in last operand of macro.
	DC	AL1 (uf)	uf = optional self-defining term and becomes user flag byte if written.
	DC	A (ccwa)	ccwa = address of first CCW to be referenced by this CCB.
	DC	7XL4'0'	Reserved for Executive to store remaining CCB information (see page 1-90).

FCP MACRO EXPANSION

EXCP

Name	Operation	Operand	Remarks
	CNOP	2,4	Needed only if location counter is not on a halfword boundary.
Name	SVC	12	
	DC	A(ccba)	Contains the address of the CCB indicating the I/O operation to be initiated.

EXCPW

Name	Operation	Operand	Remarks
	CNOP	2,4	Needed only if location counter is not on a halfword boundary.
Name	SVC	11	
	DC	A(ccba)	Contains the address of the CCB indicating the I/O operation to be initiated.

WAIT

Name	Operation	Operand	Remarks
	CNOP	2,4	Needed only if location counter is not on a halfword boundary.
Name	TM	ccba+35,X'80'	Name of the CCB indicating the I/O operation whose completion is required.
	BO	*+10	
	SVC	13	
	DC	A(ccba)	Contains the address of the CCB indicating the I/O operation whose completion is required.

FCP MACRO EXPANSION

CHECK

Name	Operation	Operand	Remarks
Name	TM	ccba+35,X'80'	ccba = name of the CCB to be checked for completion.
	BZ	inca	inca = address to which control is transferred if the order is not complete.

CPCI

Name	Operation	Operand	Remarks
Name	TM	ccba+35,X'10'	ccba = address of CCB currently being executed.
	BZ	inca	inca = address to which control is transferred if the CCB does not show that a PCI has occurred.

CCW

The CCW is a single instruction rather than a macro instruction. Therefore, the CCW is not expanded into a series of other instructions. It is encoded by the Assembler into one 8-byte machine instruction and is automatically aligned on a double word boundary.

Byte	Contents
0	Command code.
1,2,3	Memory address to or from which data is to be transferred or the memory address of the next CCW if the command is a transfer in channel.
4	Flag bytes.
5	Must be zero.
6,7	Byte count to be transferred by this operation.

FCP MACRO EXPANSION

Core Requirement Summary Table

		Required Number of Bytes
File Descriptor Macros	DTFPH	
	Magnetic Tape	192
	Direct Access	264
	DTFSR	
	Magnetic Tape	208
	Direct Access	208 to 544
	Card Reader	192
	Card Punch	192 to 352
	Printer	196
	Paper Tape Reader/Punch	192
	DTFDA	
	Direct Access	192 +
	DTFEN	8
I/O Control Macros	OPEN	24
	LBRET	6
	CLOSE	20-80
	GET	24
	PUT	24
	RELSE	24
	TRUNC	24
	CNTRL	24
	PRTOV	24
	FEOV	20-80
	READ	22
	WRITE	22
	WAITF	22
	CNTRL (SEEK)	22
	CCB	40
	EXCP	6
	EXCPW	6
	WAIT	14
	CHECK	8
	CPCI	8
	CCW	8

2. EXECUTIVE COMMUNICATION MACROS

GENERAL

◆ The Executive, located in lower memory, operates in Processor States 2, 3, and 4. It controls the system by resolving commands from the operator, interrupts from devices, and calls from user programs. These calls are written as Executive Communication Macros, and cause an interrupt to State 3, where they are analyzed and scheduled for execution.

The Executive Communication Macros and a brief statement of their use are as follows:

TYPE - enables the program to communicate with the operator via the 70/97 Console Typewriter.

COMTY - is used to insure that a previously issued typewriter request has been completed.

LPOV - is issued when the program requires that another program segment be loaded for execution. See also LPOVR.

TERMS - is used at the completion of a program when a successor program is to be loaded and initiated.

TERM - is used at the completion of a program to indicate that the job is finished.

TERMD - is used to dump the program to magnetic tape when abnormal termination is necessary.

STXIT - enables the program to specify to the Executive the addresses of one or more of the program's interrupt routines.

ERXIT - is issued at the end of a programmer supplied unrecoverable error routine to return control to the specified address.

EXIT - is issued at the end of a program's interrupt routine to return to the point in the program where the interrupt occurred.

ADEXT - enables the program to access information stored in the Executive Communication Region and the Program Table entry for this program.

DMODE - enables the program to store the 7-channel tape control byte for a given symbolic unit.

DTYPE - enables the program to store the device type for a given symbolic unit.

DDEV - is used to return a device to the system.

ASCII - enables the program to set the machine code to ASCII.

EBCD - enables the program to set the machine code to EBCDIC.

GENERAL (Cont'd)

GETOD - enables the program to get the current time of day.

SETIC - sets the user's time clock to a specified value.

GEPRY - obtains the elapsed program time.

CKPT - enables the program to write checkpoint records to a magnetic tape.

ASSGN - is used to cause device assignment to take place for a specified device.

SMODE - is used to change the 7-channel tape control byte.

QUIET - causes all outstanding I/O operations for a program to be completed before continuing.

TOCOM - enables a program to move data to the comman data area.

EXCOM - enables a program to get data from the common data area.

LPOVR - initiates the loading of a program overlay and immediately returns control to the program. (TDOS only).

FLOAT - loads a program overlay into an absolute memory location and returns control to the program after the load (TDOS only).

FLODR - initiates the loading of a program overlay into an absolute memory location and immediately returns control to the program (TDOS only).

WTOV - waits the program until the overlay requested in the LPOVR or FLODR macros are loaded (TDOS only).

STDXC - specifies to the Executive the address of the CCB to handle the Data Exchange Control and the address of the user DXC routine (TDOS only).

DXCXT - is issued at the end of a user DXC routine to return to the point in the program where the interrupt occurred (TDOS only).

SETDC - notifies the Executive that a direct control trunk is being assigned to a program (TDOS only).

DCCB - defines the direct control trunk to be handled by the direct control routine (TDOS only).

RELDC - is issued to release or deallocate a direct control trunk that was previously assigned by a SETDC macro (TDOS only).

WRTDC - transmits one byte of information over one or more direct control trunks (TDOS only).

DCWT - is issued to wait a program after a WRTDC macro is issued (TDOS only).

DCXT - is issued at the end of a user direct control routine to return to the point in the program where the interrupt occurred.

LOADP - permits a program to initiate another program internally without being terminated itself (TDOS only).

UPR - permits a program to give up control to the program below it in the Operation List without waiting itself (TDOS only).

TYPE Typewriter Requests

GENERAL DESCRIPTION

◆ The TYPE macro enables the program to communicate with the operator via the Model 70/97 Console Typewriter. Output messages may not exceed 127 bytes and reply messages from the operator may not exceed 70 bytes.

FORMAT

◆ The format of the TYPE macro is as follows:

Name	Operation	Operand
	TYPE	Message Name [,n] [, Reply-name] [, m]

SPECIFICATION RULES

Name Field

◆ Not required.

Operation Field

◆ TYPE.

Operand Field

◆ *Message-name* is a required entry that specifies the symbolic name associated with the address of the message constant to be typed.

The optional *n* entry is a decimal self-defining term indicating the length of the message and may not exceed 127. If the n entry is omitted, the length attribute assigned to the message constant is used as the length of the message.

Reply-name is an optional entry but must be included if a reply to the message is required. If included, it specifies the symbolic name associated with the address of the reply storage area.

The optional *m* entry is a decimal self-defining term indicating the length of the reply and may not exceed 70. If the m entry is omitted, the length attribute assigned to the reply storage area is used as the length of the reply.

Note: When using a common output area for TYPE commands, it is the user's responsibility to make sure the previous command has finished by issuing a COMTY command. This is to prevent the user from overlaying his first message in memory before it is typed.

COMTY Complete a Previous Type Request

GENERAL DESCRIPTION

♦ The COMTY macro is used to insure that the last typewriter request initiated by the program by using the TYPE macro has been completed. Only one typewriter request per program can be outstanding at a given time.

FORMAT

♦ The format of the COMTY macro is as follows:

Name	Operation	Operand
	COMTY	

SPECIFICATION RULES

Name Field

♦ Not required.

Operation Field

♦ COMTY.

Operand Field

♦ Not used.

General Specifications

♦ When all of the typewriter requests for the program have been completed, control is transferred to the instruction following this macro.

LPOV Load Program Overlay

GENERAL DESCRIPTION

♦ The LPOV macro is issued when the program requires that another program segment be loaded for execution. When this macro is executed the load library is positioned at the initial load header (if required), the overlay header is located, and the overlay is loaded and relocated.

FORMAT

♦ The format of the LPOV macro is as follows:

Name	Operation	Operand
	LPOV	Load-name [,address]

SPECIFICATION RULES

Name Field

♦ Not required.

Operation Field

♦ LPOV.

Operand Field

♦ *Load-name* is a required entry that specifies the name of the program segment to be loaded and can be from 1-6 characters long.

Address is an optional entry that specifies the symbolic name of the address to return to after the segment has been loaded. If this entry is omitted, control is returned to the instruction immediately following the LPOV macro.

Note: The program is placed in a Wait state until the segment is loaded.

TERMS Start Successor Program

GENERAL DESCRIPTION

♦ The TERMS macro is issued at the completion of a program when a successor program is to be initiated. All devices that are not to be used by the successor program should be deallocated (by using the DDEV macro) before this macro is issued. Otherwise, they are not returned to the system unless deallocated by the operator or the system is reloaded. The successor program must be assigned the same memory and the same priority as the calling program. Any devices that were assigned and not deallocated by the calling program are assigned to the successor program.

FORMAT

♦ The format of the TERMS macro is as follows:

Name	Operation	Operand
	TERMS	Name

SPECIFICATION RULES

Name Field

♦ Not required.

Operation Field

♦ TERMS.

Operand Field

♦ The *Name* entry specifies the name of the successor program and can be from 1-6 bytes long.

TERM Terminate Program

GENERAL DESCRIPTION

♦ The TERM macro is issued at the end of a program to inform the system and the operator that the job is finished.

FORMAT

♦ The format of the TERM macro is as follows:

Name	Operation	Operand
	TERM	

SPECIFICATION RULES

Name Field

♦ Not required.

Operation Field

♦ TERM.

Operand Field

♦ Not used.

General Specifications

♦ When this macro is executed, the following occurs:

1. All devices assigned to the program are deallocated.
2. Memory assigned to the program is deallocated.
3. The Operation List and Program Table entries are marked unassigned.
4. The operator is notified of the termination.

Device List - Bytes 8-30 for each deallocated device are cleared.

Operation List - The first byte is made $(FF)_{16}$.

Program Table entry - Program base address (four bytes) is set to zeros.

TERMD Terminate Program and Dump

GENERAL DESCRIPTION

♦ The TERMD macro is issued at the end of a program to inform the system and the operator that the job is finished. Execution of this macro also causes the program to be dumped to a magnetic tape. The operator is requested to specify the magnetic tape to which the program is to be dumped.

FORMAT

♦ The format of the TERMD macro is as follows:

Name	Operation	Operand
	TERMD	

SPECIFICATION RULES

Name Field

♦ Not used.

Operation Field

♦ TERMD.

Operand Field

♦ Not used.

General Specifications

♦ When this macro is executed, the following occurs:

1. Scratch-Pad Memory, the Executive, the program's Program Table and Device List entries, and the program including its Executive Storage Area and Run Time Parameter Area (if any) are dumped to magnetic tape.

2. The program is terminated as follows:

   a. All devices assigned to the program are deallocated.

   b. Memory assigned to the program is deallocated.

   c. The Operation List and Program Table entries are marked unassigned.

   d. The operator is notified of the termination.

Device List - Bytes 8-30 for each deallocated device are cleared.

Operation List - The first byte is made $(FF)_{16}$.

Program Table entry - Program base address (four bytes) is set to zeros.

STXIT
Set Contigency Routine Address

GENERAL DESCRIPTION

◆ The STXIT macro enables the program to specify to the Executive the addresses of one or more of the program's interrupt routines. Based on the type of interrupt condition, the Executive automatically branches to the address of the appropriate routine.

FORMAT

◆ The STXIT macro must be in the program's execution path. When the STXIT is executed the addresses of the user's routines are stored by the Executive so that when the given condition occurs a branch to the user's routine takes place.

The format of the STXIT macro is as follows:

Name	Operation	Operand
	STXIT	[n], [pc-name], [it-name] , [oc-name], [er-name], [s]

SPECIFICATION RULES

Name Field

◆ Not required.

Operation Field

◆ STXIT.

Operand Field

◆ The *n* entry is used for POS source language compatibility. It is ignored by the TOS STXIT macro. If the n entry or any other entry is missing, a comma must be entered to indicate the omission if any succeeding operands are missing.

The *PC-name* entry specifies the symbolic name associated with the address of the user's program check interrupt routine or the keyword CLOSE or ABORT.

The *IT-name* entry specifies the symbolic name associated with the address of the user's interval timer interrupt routine or the keyword CLOSE.

The *OC-name* entry specifies the symbolic name associated with the address of the user's operator communication interrupt routine or the keyword CLOSE.

The *ER-name* entry specifies the symbolic name associated with the address of the user's unrecoverable error routine or the keyword CLOSE.

The *s* entry specifies that the special termination procedures are to be followed. These procedures restrict program termination to an operator command or the user's unrecoverable error routine, ER-name. *This operand is applicable to TDOS only.*

General Specifications

◆ *Program Check Interrupt* - The following conditions cause a program check interrupt to occur:

Operation Code Trap	Exponent Underflow
Exponent Overflow	Decimal Overflow
Divide Error	Fixed-Point Overflow
Significance Error	Data Error

General Specifications *(Cont'd)*

When a program check interrupt occurs, the Executive does one of the following:

1. If the program has not specified an address of a program check routine or if the keywords CLOSE or ABORT were specified, a message is typed to the operator indicating the error and the job is terminated. The operator is given the option of dumping the program before it is terminated.

2. If the program has specified an address of a program check routine:
   a. The Executive stores the P_1 counter in the program's Executive Storage Area.

   b. The Executive stores the contents of General Registers 10 and 11 in the program's Executive Storage area. These registers are available for the program check routine to use.

   c. The interrupt weight code (indicating the specific interrupt that occurred) is stored in the program's Executive Storage Area (bytes 116-119). One of the following weight codes is stored:

Weight Code (Hexadecimal)	Meaning
00000058	Operation Code Trap.
00000064	Exponent Overflow.
00000068	Divide Error.
0000006C	Significance Error.
00000070	Exponent Underflow.
00000074	Decimal Overflow.
00000078	Fixed-Point Overflow.
00000060	Data Error.

   d. Control is transferred to the program check routine address specified.

   Note: The program check routine must terminate with an EXIT macro containing a PR operand entry. This causes the P_1 counter and General Registers 10 and 11 to be restored to the values previously stored in the Executive Storage Area.

Interval Timer Interrupt - When an interval timer interrupt occurs, the Executive does one of the following:

1. If the program has not specified an address of an interval timer routine or the keyword CLOSE was specified, a message is typed to the operator indicating the error and the job is terminated. The

General Specifications *(Cont'd)*

operator is given the option of dumping the program before it is terminated.

2. If the program has specified an address of an interval timer routine:

   a. The Executive stores the P_1 counter in the program's Executive Storage Area.

   b. The Executive stores the contents of General Registers 10 and 11 in the program's Executive Storage Area. These registers are available for the interval timer routine to use.

   c. Control is transferred to the interval timer routine address specified.

Note: The interval timer routine must terminate with an EXIT macro containing a TR operand. This causes the P_1 Counter and General Registers 10 and 11 to be restored to the values previously stored in the Executive Storage Area.

Operator Communication Interrupt - When an operator communication interrupt occurs, (the operator presses the COIN key and types E INT program number), the Executive does one of the following:

1. If the program has not specified an address of an operator-communication routine or the keyword CLOSE was specified, the interrupt is ignored.

2. If the program has specified an address of an operator communication routine:

   a. The Executive stores the P_1 Counter in the program's Executive Storage Area.

   b. The Executive stores the contents of General Registers 10 and 11 in the program's Executive Storage Area. These registers are available as base registers for the operator-communication routine. If additional registers other than 10 and 11 are required, the original contents of those registers must be saved prior to their use and restored after their use and prior to exiting from the operator-communication routine.

   c. Control is transferred to the operator communication routine address specified.

Notes: 1. The TYPE macro can be used to communicate between the program and the operator.

2. The operator-communication routine must terminate with an EXIT macro containing a CR operand. This causes the P_1 counter and General Registers 10 and 11 to be restored to the values previously stored in the Executive Storage Area.

3. If the operator attempts to interrupt a program that is already servicing an operator communication (an EXIT-CR macro has not been issued), the interrupt is ignored.

Executive Procedures for Real Time Error Handling

◆ The user, who is processing in an environment where a sudden and arbitrary cessation cannot be tolerated (e.g., "on line" and communications applications), is provided with a set of TDOS procedures to permit options by which he can recover from the situation causing the cessation or, at least, of terminating the processing in an orderly manner.

The algorithm for recovery or orderly termination must be contained in the user contingency routine for unrecoverable errors as defined by the STXIT macro (s optional parameter) to indicate that the special procedures are to be employed. When either an unrecoverable error or a doubtful termination occurs, control is given to the user's unrecoverable error routine. This coding, by examining the information contained in the ESA, may validate the termination or specify that control is to be returned to his main routine at a definite point.

The only way in which a program operating under these procedures can be terminated is (1) by operator command or (2) while in its unrecoverable error routine. Any other termination request is detected when the program is otherwise eligible to run (i.e., is not waited) and is cancelled, with the user regaining control as described above.

Note:

It should be noted that any error occurring after the cancellation of the termination and before the user gains control at the address specified by the ERXIT macro results in a termination of his program just as if these procedures had not been requested.

Reason Codes

◆ When the special termination procedures are specified by a valid STXIT macro, they are in effect until cancelled by a subsequent valid STXIT. Control then is given to the unrecoverable error routine in two general cases:

1. When an interrupt designated as an unrecoverable error (normally Privileged Operation or Address Error) is encountered. The interrupt weight is stored in bytes 116-119 of the user's ESA. For practical purposes, the weight may be said to be stored in byte 119, a distinction which can be of importance for analysis. This condition is the only one which can cause entry to the routine in the absence of special procedure specification.

2. When special procedures have been specified control is given to the unrecoverable error routine itself. The reason for termination, corresponding to the reason code which would normally be displayed on the console typewriter for the operator's response, is stored in byte 116 of the user's ESA. The user's analysis routine may then use this information, the stored P-counter, information stored internally in the program, etc, to determine the action required.

Executive Procedures for Real Time Error Handling *(Cont'd)*

Regardless of the reason for entering the unrecoverable error coding and also of whether special termination procedures are in effect, a user may elect to conclude his unrecoverable error processing. This error processing may be caused by either a valid termination (normally via a TERM, TERMD, or TERMS although generation of any condition which normally causes termination would suffice) or by issuing an ERXIT macro with a valid branch address. In this latter case, all contingency routines are made available (all registers saved for other contingencies, except DC and DXC, are lost). GR10P1 and GR11P1 are restored to the values they had immediately prior to entry into the unrecoverable error routine, and control is given to the main path at the location specified by the operand field. Furnishing an invalid address in the operand field is equivalent to validating the termination, i.e., the user program is terminated.

It is the responsibility of the user to assure that the required information to perform his unrecoverable error analysis when the reason code and program counter are insufficient, is stored internally in the user's program area. It is also his responsibility to provide for resumption of processing of DC and DXC contingencies where appropriate.

The standard format of TDOS Executive message is as follows:

n pppppp idcc [A or Δ]

n = Program Identification Code.

pppppp = Program Name.

id = Component ID:

- 00 - Error Recovery.
- 01 - Executive I/O.
- 02 - Executive (all other).
- 03 - Monitor.

cc = Reason Code:

Reason codes and their definitions are described on the following page.

A = Response Required.

Δ = No response Required.

Reason Codes (Cont'd)

Shown below are the "cc" Reason Codes:

Reason Codes	Definition
11	CCB address or instruction error.
12	Device list address error. Assignment halfword does not address the first byte of a device list entry.
13	SVC calling sequence error.
15	Program error in CCW: data or protection key error.
16	Device mn inoperable during execution or program will not accept unrecoverable errors.
17	Invalid SVC.
18	Invalid interrupt.
19	Last user instruction caused an interrupt (priority 22 through 31) for which no STXIT was executed.
20	Unrecoverable read error from disc while reading in Executive overlay.
21	Assignment for required device was NO.
27	Monitor subprocessor or successor not found.
28	Load address too high, Monitor subprocessor or successor program.
29	Unrecoverable error searching for PD or Program not found.
30	Load exceeds allocated memory.
32	Insufficient memory (tape loading).
33	LD entry or load not found for user program segment.
35	Unrecoverable read error loading user program or device inoperable.
36	User Load exceeds memory limit or FLOAT or FLODR call asking to be loaded outside of program area.
50	Invalid user key-in address in TYPE macro.

TDOS FCP Procedures for Real Time Error Handling

◆ The user now has the ability to gain control in a TDOS real time communications environment when a potentially nonrecoverable error occurs in FCP. He may analyze the actual typeout message in memory, if such is the case; or, if an automatic TERMD was issued, a two-byte reason code follows that SVC. If, after determining the nature of the error, and that it is recoverable, he can insert the proper reply into the reply area pointed to by the original type expansion and continue or, if necessary he can close his lines down orderly and then issue his own TERMD. The following requirements are necessary:

1. The user must specify the s parameter along with the ER-name error routine name parameter in the STXIT macro at source time.
   a. When the s parameter is used, the FCP, at object time modifies the TYPE SVC to a TERMD SVC. This provides a unique interface with the Executive system which in turn gives control to the user.

2. In the user's error routine, the user must determine where the P counter is pointing.
   a. If the Address of the ESA is known, bytes 8-11 of the ESA will contain the P counter address.
   b. If the ESA address is not known, an ADEXT macro must be issued. Register 15 is loaded with the address of the program table entry. Bytes 24-27 of the program table entry contain the ESA address.
      (1) The user must insure the integrity of Registers 0, 1, 14, and 15 as in the past.
      (2) Any other registers which are required by the user's program such as IOREG should maintain their integrity while in the user's error routine.
   c. Within the ESA (bytes 172-175) is stored the P counter of an unrecoverable error interrupt.

3. The P counter points to the byte following the TERMD SVC which would be either a message length indicator for the TYPE expansion or an alphabetic reason code for the TERMD. The user must determine which.
   a. A non zero numeric byte represents an altered TYPE SVC. (See Example 1.)
      (1) When this condition exists, the next three bytes (Program Counter +1 thru 3) contain the address of what normally would have been the typed out error message.

TDOS FCP Procedures for Real Time Error Handling *(Cont'd)*

(2) The format of this error message is as follows:

CCCCR FFFFFF NN

where: CCCC is the four-digit error code.

R is a one-byte indicator indicating that a reply is necessary. This will be an A.

FFFFFF is the file name

NN is the physical device assigned.

(3) The user must interrogate the error code which is documented in the TDOS Operator's Guide (70-35-404).

(4) The user must place his desired response in the location specified by the address contained in Program Counter +5 thru 7.

(5) To continue processing, the user must use the ERXIT macro with the operand pointing to the instruction located at Program Counter +8. If the user wishes to terminate, use the TERMD inside of the error routine.

b. Two alphabetic bytes following the TERMD SVC indicates the existance of an original TERMD SVC. (See Example 2.)

(1) These two bytes are the appropriate reason codes.

(a) AA reason code is issued when a FEOV is used to a non-tape device.

(b) BB reason code is issued when the user provides no ERROPT or WLRERR in his file description to support error conditions.

(c) CC reason code is issued when an abnormal termination of an I/O occurs and, thru various software tests, is found to be caused by hardware.

(2) In these cases, there is no logical return to FCP.

c. Random Access FCP, on certain errors, produces an error message with no reply followed by an automatic TERMD.

(1) The absence of the R character in the message area plus the message number denotes this condition.

(2) The user should then issue the TERMD macro, when applicable, within the error routine.

Example 1

◆ Type SVC Expansion

```
0A01        SVC     01 (I converted to TERMD SVC)
90          DC      X'90'
000476      DC      AL3 (message location)
01          DC      X'01'
00034D      DC      AL3 (message reply)

            CL16'50CCA FFFFFF  NN'
```

Example 2

◆ TERMD SVC Expansion

0A16	SVC	22	TERMD
C1C1	DC	X'C1C1'	Reason Code

ERXIT Exit From Unrecoverable Error Routine

GENERAL DESCRIPTION

◆ The ERXIT macro is issued at the end of a programmer supplied unrecoverable error routine to return control to the specified address.

FORMAT

◆ The format of the ERXIT macro is as follows:

Name	Operation	Operand
	ERXIT	Return Name

SPECIFICATION RULES

Name Field

◆ Not required.

Operation Field

◆ ERXIT.

Operand Field

◆ *Return Name* specifies the symbolic name of the location to which control is to be returned, i.e., the first location of non-contingency coding to be executed.

Notes:

Execution of this macro restores GR10P1 and GR11P1 to their status before the unrecoverable error code was entered, sets the P1 Program Counter to the address specified by Return name, and *cancels all indication of existing contingency routines (except DC and DXC) in use.* That is, if a contigency routine other than DC or DXC has been entered and a corresponding EXIT macro has not been executed, the TDOS Executive cancels its internal flags indicating that the EXIT is yet to be encountered. The saved GR10P1 and GR11P1 and program counter for such contigency routine(s) are lost and a subsequent execution of the EXIT macro without a reoccurrence of the condition causing entry into the contingency routine is treated as an invalid SVC.

General Specifications *(Cont'd)*

Unrecoverable Error Interrupt - The following conditions cause an unrecoverable error interrupt to occur:

Privileged Operation

Address Error

When an unrecoverable error interrupt occurs, the Executive does one of the following:

1. If the program has not specified an address of an unrecoverable error interrupt routine or the keyword CLOSE was specified, a message is typed to the operator indicating the error and the job is terminated. The operator is given the option of dumping the program before it is terminated.

2. If the program has specified the address of an unrecoverable error routine:

   a. The Executive stores the P_1 Counter in the program's Executive Storage Area.

   b. The Executive stores the contents of General Registers 10 and 11 in the program's Executive Storage Area. These registers are available for the unrecoverable error routine to use.

   c. The interrupt weight code (indicating the specific interrupt that occurred) is stored in the program's Executive Storage Area (bytes 116-119). One of the following weight codes is stored:

Weight Code (Hexadecimal)	Meaning
00000054	Privileged Operation.
0000005C	Address Error.

   d. Control is transferred to the unrecoverable error routine address specified.

Note: The unrecoverable error routine must end with a TERM or a TERMD macro.

EXIT
Exit

GENERAL DESCRIPTION

♦ The EXIT macro is issued at the end of a programmer-supplied interrupt routine to return to the point in the program where the interrupt occurred.

FORMAT

♦ The format of the EXIT macro is as follows:

Name	Operation	Operand
	EXIT	PR, TR, or CR

SPECIFICATION RULES

Name Field

♦ Not required.

Operation Field

♦ EXIT.

Operand Field

♦ One of the following entries can appear in this field:

PR - Return from the program's program check routine.

TR - Return from the program's timer routine (for timer interrupt).

CR - Return from the program's operator-communication routine (operator-communication interrupt).

ADEXT Address Executive Tables

GENERAL DESCRIPTION

♦ The ADEXT macro enables the program to address information stored in the Executive Communication Region and the Program Table Entry for this program.

FORMAT

♦ The format of the ADEXT macro is as follows:

Name	Operation	Operand
	ADEXT	

SPECIFICATION RULES

Name Field

♦ Not Required.

Operation Field

♦ ADEXT.

Operand Field

♦ Not used.

General Specifications

♦ When this macro is issued, the following P_1 (Processing State) registers are loaded:

General Register 14 - Contains the address of the first byte of the Executive Communication region.

General Register 15 - Contains the address of the first byte of the Program Table Entry for this program.

Note: The address in bytes 24-27 of the Program Table Entry can be used to obtain information contained in the Executive Storage Area for this program.

The contents of the Executive Communication region, the Program Table entry, and the Executive Storage area are listed in the TOS Control System Reference Manual, No. 70-00-609.

DMODE Access Tape Control

GENERAL DESCRIPTION

♦ The DMODE macro enables the program to obtain the 7-channel tape control byte for a given symbolic device.

FORMAT

♦ The format of the DMODE macro is as follows:

Name	Operation	Operand
	DMODE	nnnnnn,R,W

SPECIFICATION RULES

Name Field

♦ Not required.

Operation Field

♦ DMODE.

Operand Field

♦ The *nnnnnn* entry specifies the six-byte symbolic name of the device.

The *R* entry is used for POS source language compatibility. If the R entry is missing, a comma must be entered to indicate the omission. It is ignored by the TOS DMODE macro.

The *W* entry is the symbolic name of the one-byte location in which the control byte for the device named is to be stored.

General Specifications

♦ The possible control bytes are listed under the SMODE macro.

DTYPE
Access Device Table

GENERAL DESCRIPTION

◆ The DTYPE macro enables the program to obtain the device type for a given symbolic unit.

FORMAT

◆ The format of the DTYPE macro is as follows:

Name	Operation	Operand
	DTYPE	nnnnnn,R,W

SPECIFICATION RULES

Name Field

◆ Not required.

Operation Field

◆ DTYPE.

Operand Field

◆ The *nnnnnn* entry specifies the six-character symbolic name of the device.

The *R* entry is used for POS source language compatibility. If the R entry is missing, a comma must be entered to indicate the omission. It is ignored by the TOS DTYPE macro.

The *W* entry is the symbolic name of the one-byte location in which the device type code for the device named is to be stored.

Device assignment must be made prior to the execution of this macro.

General Specifications

◆ The device type codes are as follows:

Device Type Code (Hexadecimal)	Device
00	70/97 Console Typewriter
01	70/242 Printer
02	70/243 Printer
03	70/248 Bill Feed Printer
04	70/234 Card Punch
05	70/236 Card Punch
06	70/237 Card Reader
07	70/221 Paper Tape Reader/Punch
08	70/251 Videoscan Document Reader
0A	7-Channel Tape Station

General Specifications *(Cont'd)*

Device Type Code (Hexadecimal)	Device
0B	9-Channel Tape Station
0C	70/564 Disc Storage Unit
0D	70/565 Drum Memory Unit-32 Cylinders
1D	70/565 Drum Memory Unit-64 Cylinders
2D	70/565 Drum Memory Unit-96 Cylinders
3D	70/565 Drum Memory Unit-128 Cylinders
4D	70/565 Drum Memory Unit-160 Cylinders
5D	70/565 Drum Memory Unit-192 Cylinders
6D	70/565 Drum Memory Unit-244 Cylinders
7D	70/565 Drum Memory Unit-256 Cylinders
0F	70/568 Mass Storage Unit

DDEV Deallocate Device

GENERAL DESCRIPTION

♦ The DDEV macro is used to return a device to the system. This macro should be given after the last I/O has been issued to the device.

FORMAT

♦ The format of the DDEV macro is as follows:

Name	Operation	Operand
	DDEV	nnnnnn

SPECIFICATION RULES

Name Field

♦ Not required.

Operation Field

♦ DDEV.

Operand Field

♦ The *nnnnnn* entry specifies the symbolic name of the device to be deallocated.

General Specifications

♦ When this macro is executed, the specified device is made available to the system.

ASCII Set ASCII Mode

GENERAL DESCRIPTION

♦ The ASCII macro enables the program to set the machine code to USASCII.

FORMAT

♦ The format of the ASCII macro is as follows:

Name	Operation	Operand
	ASCII	

SPECIFICATION RULES

Name Field

♦ Not required.

Operation Field

♦ ASCII.

Operand Field

♦ Not used.

EBCD Set EBCDIC Mode

GENERAL DESCRIPTION

♦ The EBCD macro enables the program to set the machine code to EBCDIC.

FORMAT

♦ The format of the EBCD macro is as follows:

Name	Operation	Operand
	EBCD	

SPECIFICATION RULES

Name Field

♦ Not required.

Operation Field

♦ EBCD.

Operand Field

♦ Not used.

SETIC
Set Time Clock

GENERAL DESCRIPTION

◆ The SETIC macro is used to set the user program's time clock to a specified value. This value is constantly decremented until it reaches zero, at which time an interrupt occurs. The Executive resets the user program's time clock to this value and transfers the user program's STXIT TR address. If the user program has not specified a STXIT TR address it is terminated when the elapsed time clock interrupt occurs.

FORMAT

◆ The format of the SETIC macro is as follows:

Name	Operation	Operand
	SETIC	ptc

SPECIFICATION RULES

Name Field

◆ Not required.

Operation Field

◆ SETIC

Operand Field

◆ ptc is the symbolic name of the user's six-byte area that contains the value to be set into the user program's time clock.

GEPRT
Get Program Time

GENERAL DESCRIPTION

- The GEPRT macro is used to obtain elapsed program time.

FORMAT

- The format of the GEPRT macro is as follows:

Name	Operation	Operand
	GEPRT	prt

SPECIFICATION RULES

Name Field

- Not required.

Operation Field

- GEPRT

Operand Field

- prt is the symbolic name of the user's six-byte area to receive the program time in the zoned decimal form (HHMMss).

GETOD Get Time of Day

GENERAL DESCRIPTION

◆ When the system is loaded, the operator is asked to type the time of day. The Executive maintains the current time throughout the day. The GETOD macro is used to get the current time of day.

FORMAT

◆ The format of the GETOD macro is as follows:

Name	Operation	Operand
	GETOD	tod

SPECIFICATION RULES

Name Field

◆ Not required.

Operation Field

◆ GETOD.

Operand Field

◆ *tod* - is the symbolic name of the user's six-byte time of day storage area in which the Executive places the current time of day. The time of day storage area has the following format:

HHMMSS

where:

HH = Hours
MM = Minutes
SS = Seconds

Each byte is in zoned decimal (character)form. For example:

9:10 AM appears as: F0F9F1F0F0F0 at location tod.

Operand Field *(Cont'd)*

The third operand, *checkpoint ID*, is a unique five-byte identification for a set of checkpoint records. The checkpoint ID is written to the Checkpoint Header Record and is used at restart time in order to locate this set of checkpoint records. If there is more than one checkpoint taken in a program, it is the user program's responsibility to update the checkpoint ID in the CKPT expansion for each set of checkpoint records. The characters in the checkpoint ID can be any combination of the following: A to Z, 0 to 9, ¤ , or blank.

The fourth operand, *DTF-address*, is the symbolic name of the first DTF (DTFSR, DTFDA, or DTFPH) for this program that the user wants positioned and label checked at restart time. All DTF's that do not require positioning and label checking must precede this DTF. If this operand is omitted, no positioning or label checking is done at restart time.

The fifth operand, *disc-address pointer* is the location on disc where the checkpoint records are to be written. *This entry is applicable to TDOS only.* If this operand is present the checkpoint records are written to disc; if not, on tape. This operand points to an address in the user program of an eight-byte field.

1. The left-hand four bytes of the field is the disc/drum address (CCHH - Cylinder/Head) at which this set of checkpoint records is to be stored.
2. The right-hand four bytes of the field is the disc/drum address (CCHH - Cylinder/Head) which is the end boundary for this set of checkpoint records.

General Specifications

◆ 1. Any or all of the operands can be omitted, but commas must be written to indicate omitted operands.

2. If the checkpoint device is a tape under logical level FCP control, this file must be opened before the CKPT macro is executed. In TDOS, if the checkpoint device is a disc or drum, the file must be opened prior to the CKPT macro.

3. If the checkpoint device is a tape under physical level FCP control, the user program must issue an ASSGN macro to the checkpoint device before the CKPT macro provided that the device has not been assigned previously.

4. The user program must inform FCP of the device to which the checkpoint records are to be written. This is done by the user at program execution time by moving the assignment halfword (bytes 14 and 15) from the DTF expansion (if at logical level FCP) or the CCB (if at physical level FCP) into the checkpoint expansion (bytes 30 and 31).

General Specifications *(Cont'd)*

Suggested methods for the above specifications:

Logical level FCP

```
CHECK    DTFSR
           .
           .
           .
           .
         DTFEN
           .
           .
           .
           .
         OPEN      CHECK
           .
           .
           .
           .
ACON     DC        A(CHECK+14)     Address constant
           .                        of assignment
           .                        halfword.
           .
           .
         L         n,ACON          Move address
           .                        of assignment
           .                        halfword into
           .                        a register.
           .
         MVC       POINT+30(2),0(n)
           .                        Move assign-
           .                        ment halfword
           .                        into byte 10 of
           .                        CKPT ex-
POINT    CKPT                     pansion
```

Physical level FCP

```
CHECK    CCB
           .
           .
           .
           .
         ASSGN     CHECK
           .
           .
           .
           .
ACON     DC        A(CHECK+14)
           .
           .
           .
           .
         L         n,ACON
         MVC       POINT+30(2),0(n)
           .
           .
           .
           .
POINT    CKPT
```

Note: n = any register

CKPT Checkpoint

GENERAL DESCRIPTION

◆ The CKPT covers checkpoint records to be written to a magnetic tape in TOS, or to a disc or magnetic tape in TDOS. These records contain the status of the program and the system at the time that the CKPT is issued. They provide a means of restarting at a midway point in the program rather than at the beginning. The restarting of programs for which checkpoint records have been written is initiated by the operator. If checkpoint records are to be written to a tape, they can be written either to a tape used solely for checkpoint dumping or to an output tape containing data records.

FORMAT

◆ The format of the CKPT macro is as follows:

Name	Operation	Operand
Blockname	CKPT	restart-name
		error routine-name,
		checkpoint ID,
		DTF-address,
		disc-address pointer

SPECIFICATION RULES

Name Field

◆ The *Blockname*, is required and is the symbolic name associated with the first byte of the expansion.

Operation Field

◆ CKPT.

Operand Field

◆ The first operand, *restart-name*, specifies the symbolic name of the location to which control is to be transferred after restart has been accomplished. If this operand is omitted, the address of the first byte following the CKPT expansion is generated.

The second operand, *error routine-name*, specifies the symbolic name of the location to which control is to be transferred if an unrecoverable error occurs during checkpoint. The error byte, byte 40 in the expansion, is set and a branch to error routine-name is generated. The hexadecimal error byte codes are:

01 - Checkpoint parameter error.

02 - Pack/unpack mode not being used for a seven-level checkpoint tape.

10 - Unrecoverable or abnormal termination during last user I/O on checkpoint tape.

80 - Abnormal termination during checkpoint (ET).

81 - Unrecoverable error during checkpoint.

82 - Service request not honored during checkpoint.

Special Considerations

◆ 1. If checkpoint records are to be written to tape, they can be written to either (1) an output tape solely used for checkpoint dumping or (2) an output tape, containing data records. However, checkpoint records are not counted in the block count for a data set.

2. If the checkpoint device is a seven-level tape, the pack/unpack mode must be used.

3. In TDOS, random access FCP controlled files must be reopened after restart.

4. In TDOS, if the checkpoint device is a disc, the disc overflow feature must be present.

5. Only programs which are independent of Monitor can be restarted.

6. External requests, such as console requests, are not remembered at restart time.

7. If any input data tapes have checkpoint records interspersed with data records, CKPTREC=YES in the DTFSR must be used.

8. Registers 14 and 15 must be stored by the user before CKPT and restored after CKPT if FCP is used.

9. It is assumed that all random access files are available at restart time in exactly the same state of information storage as they were at checkpoint time.

ASSGN
Assign Device

GENERAL DESCRIPTION

◆ The ASSGN macro is used to cause device assignment to take place for a specified symbolic device name.

FORMAT

◆ The format of the ASSGN macro is as follows:

Name	Operation	Operand
	ASSGN	Blockname

SPECIFICATION RULES

Name Field

◆ Not required.

Operation Field

◆ ASSGN.

Operand Field

◆ The blockname entry specifies the symbolic name associated with the CCB (Command Control Block) established for the particular device.

Operand Field *(Cont'd)*

Note: *(Cont'd)* When this macro is executed the write control byte specified in the SMODE operation is placed in the Device List entry for this device.

The write control byte in the device list entry for this device is placed in the macro expansion. In other words the two write control bytes are exchanged.

Repeated execution of the same SMODE macro has the effect of a flip-flop between the two write control bytes.

SMODE
Set Write Control Mode

GENERAL DESCRIPTION

◆ The SMODE macro is used to change the 7-channel tape control byte for a specific device. This macro can only be used after device assignment for the device concerned has taken place.

FORMAT

◆ The format of the SMODE macro is as follows:

Name	Operation	Operand
	SMODE	YY

SPECIFICATION RULES

Name Field

◆ Not required.

Operation Field

◆ SMODE.

Operand Field

◆ YY is a self-defining term that must be written as two hexadecimal characters. It specifies the Write Control mode to be established. One of the following write control bytes may be specified:

Control Byte (Hexadecimal)	Density (Bytes) In	Parity	Pack/Unpack Mode	Translate Mode
60	200	Odd	Off	Off
A0	556	Odd	Off	Off
E0	800	Odd	Off	Off
40	200	Even	Off	Off
80	556	Even	Off	Off
C0	800	Even	Off	Off
68	200	Odd	Off	On
A8	556	Odd	Off	On
E8	800	Odd	Off	On
48	200	Even	Off	On
88	556	Even	Off	On
C8	800	Even	Off	On
70	200	Odd	On	Off
B0	556	Odd	On	Off
F0	800	Odd	On	Off

Note: The program must place the assignment halfword for the device concerned into the expansion for this macro prior to issuing this macro. The assignment halfword is obtained from the CCB (bytes 14-15) generated for this device.

TOCOM
Move to Common Data Area

GENERAL DESCRIPTION

◆ The TOCOM macro is issued when a program desires to move data to the Common Data Area.

FORMAT

◆ The format of the TOCOM macro is as follows:

Name	Operation	Operand
	TOCOM	Loc, Data, Length

SPECIFICATION RULES

Name Field

◆ Not required.

Operation Field

◆ TOCOM.

Operand Field

◆ *Loc* - a decimal number indicating the location relative to the beginning (designated by 1) of the Common Data Area into which the first byte of data is to be moved.

Data - the symbolic address of the first byte in the user's program to be moved into the Common Data Area.

Length - an optional entry indicating the (decimal) number of bytes to be moved. If Length is omitted the implied length of Data is used.

EXCOM
Get From Common Data Area

GENERAL DESCRIPTION

◆ The EXCOM macro is issued when a program desires to get data from the Common Data Area.

FORMAT

◆ The format of the EXCOM macro is as follows:

Name	Operation	Operand
	EXCOM	Loc, Data, Length

SPECIFICATION RULES

Name Field

◆ Not required.

Operation Field

◆ EXCOM.

Operand Field

◆ *Loc* - a decimal number indicating the location relative to the beginning (designated by 1) of the Common Data Area from which the first byte of data is to be retrieved.

Data - the symbolic address of the distinction of the first byte to be retrieved.

Length - an optional entry indicating the (decimal) number of bytes to be moved. If Length is omitted the implied length of Data is used.

QUIET
Quiet Input/Output Devices

GENERAL DESCRIPTION

♦ The QUIET macro waits the calling program until all I/O operations in progress or on queue have been completed. When all I/O operations have been completed, control is transferred to the instruction following the QUIET macro.

FORMAT

♦ The format of the QUIET macro is as follows:

Name	Operation	Operand
	QUIET	

SPECIFICATION RULES

Name Field

♦ Not required.

Operation Field

♦ QUIET.

Operand Field

♦ Not required.

LPOVR Load Program Overlay and Return

GENERAL DESCRIPTION

- The LPOVR macro is issued when another program segment is to be loaded and control is transferred back to the program before the segment is actually loaded.

FORMAT

- The format of the LPOVR macro is as follows:

Name	Operation	Operand
	LPOVR	Load-name [,address]

SPECIFICATION RULES

Name Field

- Not required.

Operation Field

- LPOVR.

Operand Field

- *Load-name* is a required entry that specifies the 1-6 character name of the program segment to be loaded.

Address is an optional entry that specifies the symbolic name of the address to return to before the completion of the loading of the segment. If this entry is omitted, control is returned to the instruction immediately following the LPOVR macro.

FLOAT
Float Program Overlay

GENERAL DESCRIPTION

◆ The FLOAT macro is issued to load a segment into memory at an *absolute address* and return control to the program *after* the segment is loaded. It is the *user's responsibility* to see that the absolute load address is within the program area and that all ENTRY's and EXTRN's are resolved.

FORMAT

◆ The format of the FLOAT macro is as follows:

Name	Operation	Operand
	FLOAT	load-name,load-addr [,address]

SPECIFICATION RULES

Name Field

◆ Not required.

Operation Field

◆ FLOAT.

Operand Field

◆ *Load-name* is a required entry that specifies the 1-6 character name of the program segment to be loaded.

Load-addr is a required entry that specifies the absolute core address at which this segment is to be loaded.

Address is an optional entry that specifies the symbolic name of the address to return to after the completion of the loading of a segment. If this entry is omitted, control is returned to the instruction immediately following the FLOAT macro.

Notes:

1. The FLOAT macro is restricted to use with Random Access Devices.
2. Where a DS statement precedes an overlay area, the "Load Address" of a "FLOAT" macro should reflect the first text location.

FLODR Float Program Overlay and Return

GENERAL DESCRIPTION

♦ The FLODR macro is issued to load a segment into memory at an *absolute address* and returns control to the program *before* the segment is actually loaded. It is the *user's responsibility* to see that the absolute load address is within the program area and that all ENTRY's and EXTRN's are resolved.

FORMAT

♦ The format of the FLODR macro is as follows:

Name	Operation	Operand
	FLODR	Load-name, Load-addr [,address]

SPECIFICATION RULES

Name Field

♦ Not required.

Operation Field

♦ FLODR.

Operand Field

♦ *Load-name* is a required entry that specifies the 1-6 character name of the program segment to be loaded.

Load-addr is a required entry that specifies the absolute core address at which this segment is to be loaded.

Address is an optional entry that specifies the symbolic name of the address to return to before the completion of the loading of the segment. If this entry is omitted, control is returned to the instruction immediately following the FLODR macro.

WTOV
Wait for Overlay

GENERAL DESCRIPTION

♦ The WTOV macro is issued in conjunction with the LPOVR and FLODR macros. It waits the program until the segment requested in the previous LPOVR macro is loaded.

FORMAT

♦ The format of the WTOV macro is as follows:

Name	Operation	Operand
	WTOV	[address]

SPECIFICATION RULES

Name Field

♦ Not required.

Operation Field

♦ WTOV.

Operand Field

♦ *Address* is an optional entry that specifies the symbolic name of the address to return to after the load. If this entry is omitted, control is returned to the instruction immediately following the WTOV macro.

STDXC Set Address of DXC Routine

GENERAL DESCRIPTION

♦ The STDXC macro enables the program to specify to the Executive the address of the CCB to handle the DXC. At the time of an interrupt the Executive jumps to the user routine specified in the CCB.

FORMAT

♦ The format of the STDXC macro is as follows:

Name	Operation	Operand
	STDXC	CCB-name

SPECIFICATION RULES

Name Field

♦ Not required.

Operation Field

♦ STDXC.

Operand Field

♦ *CCB-name* is the symbolic name of the CCB which is used to reference the DXC.

General Specifications

♦ The CCB for the DXC is the same as the normal CCB with the exception that one more field is added.

The format of the CCB for the DXC is as follows:

Name	Operation	Operand
	CCB	nnnnnn, command list name [,X'YY', device class, dxc routine]

The first four fields in the Operand field of the CCB, which generate the first 40 bytes of the CCB, are the same as defined under the CCB description. The additional field, *dxc routine,* is the symbolic name of the DXC routine *supplied by the user.*

DXCXT Exit From User DXC Routine

GENERAL DESCRIPTION

♦ The DXCXT macro is issued to exit from the user's DXC routine. This macro must be issued before the Executive will allow this routine to process another DXC interrupt.

FORMAT

♦ The format of the DXCXT macro is as follows:

Name	Operation	Operand
	DXCXT	CCB-name

SPECIFICATION RULES

Name Field

♦ Not required.

Operation Field

♦ DXCXT.

Operand Field

♦ The *CCB-name* is the symbolic name of the CCB as specified in the STDXC macro.

SETDC Set Address of User's Direct Control Routine

GENERAL DESCRIPTION

◆ The SETDC macro must be issued for each direct control trunk to be handled by a program. This notifies the Executive that this trunk is being assigned to this program.

FORMAT

◆ The format of the SETDC macro is as follows:

Name	Operation	Operand
	SETDC	DCCB-name

SPECIFICATION RULES

Name Field

◆ Not required.

Operation Field

◆ SETDC.

Operand Field

◆ The *DCCB-name* is the symbolic name of the DCCB macro associated with the trunk to be handled. There must be a different DCCB macro for each trunk handled.

Notes

◆ 1. This macro must be executed for each Direct Control Trunk to be handled.

2. There must be a different DCCB for each trunk handled.

3. This macro points to a DCCB which indicates the trunk to be handled and the address of the users Direct Control routine.

4. Once a SETDC macro is executed the trunk is assigned to the program until a subsequent RELDC (Release DC trunk) macro is executed.

5. If no user routine is specified in the DCCB, the trunk is still assigned to the user. When an external interrupt occurs on this trunk, the byte transmitted is stored in the DCCB but the program is not notified.

Notes

◆ 1. A processor cannot Read Direct to itself. The I-field bit associated with the receiving processor must always be set to zero. Therefore, based upon how the Direct Control trunks are set up, one of the above bit patterns would be invalid.

2. The second byte of the expansion is used to store the input byte.

3. The last 12 bytes of the expansion are used to store the users in-line return address, BR 10, and BR 11 on entrance to the user-supplied routine.

4. When a Direct Control trunk is to be handled by a program (see SETDC macro) and the DC-routine is not specified in the DCCB expansion (no users Direct Control routine), external interrupts for this trunk will be handled by storing the input byte in the second byte of the expansion with no notification to the user.

DCCB Direct Command Control Block

GENERAL DESCRIPTION

◆ The DCCB macro defines the direct control trunk to be handled, and the address of the user's direct control routines.

FORMAT

◆ The format of the DCCB macro is as follows:

Name	Operation	Operand
	DCCB	I-field [,DC-routine]

SPECIFICATION RULES

Name Field

◆ Not required.

Operation Field

◆ DCCB

Operand Field

◆ The *I-field* is a required one-byte self-defining value which specifies one of up to five possible sets of direct control trunks to be handled by this program. The byte I-field is used as the I-field in the Read Direct instruction and must conform to one of the following patterns.

Bit Pattern								Trunk Handled
2^7	2^6	2^5	2^4	2^3	2^2	2^1	2^0	
1	0	0	0	0	0	0	0	Six
0	1	0	0	0	0	0	0	Five
0	0	1	0	0	0	0	0	Four
0	0	0	1	0	0	0	0	Three
0	0	0	0	1	0	0	0	Two
0	0	0	0	0	1	0	0	One

The *DC-routine* is an optional entry of the symbolic name of the user's direct control routine. If this entry is omitted, the DC interrupt will be processed by the Executive but control will not be given to any user routine.

RELDC Release Use of Direct Control Trunk

GENERAL DESCRIPTION

◆ The RELDC macro is issued to release a direct control trunk from a program. It must have been previously assigned by a SETDC macro.

FORMAT

◆ The format of the RELDC macro is as follows:

Name	Operation	Operand
	RELDC	DC-trunk

SPECIFICATION RULES

Name Field

◆ Not required.

Operation Field

◆ RELDC.

Operand Field

◆ The *DC-trunk* is a required one-byte self-defining value which specifies up to five possible direct control trunks to be released. (See DCCB macro for format).

Note

◆ 1. This macro can be used to release any number of Direct Control trunks being handled by a program.

WRTDC Execute Write Direct

GENERAL DESCRIPTION

◆ The WRTDC macro transmits one byte of information over one or more direct control trunks. A DCWT macro must be issued before another WRTDC macro can be issued.

FORMAT

◆ The format of the WRTDC macro is as follows:

Name	Operation	Operand
	WRTDC	DC-trunk, DC-byte

SPECIFICATION RULES

Name Field

◆ Not required.

Operation Field

◆ WRTDC.

Operand Field

◆ *DC-trunk* is a required one-byte self-defining value which specifies the trunk or trunks that one byte is to be transmitted over. (See DCCB macro for format).

The *DC-byte* is a required one-byte self-defining value which is the byte to be transmitted.

Notes

◆ 1. The byte specified by DC-trunk becomes the I-field of a Write Direct instruction.

2. Once a WRTDC macro is issued a DCWT macro must be issued before another WRTDC is attempted.

3. The trunks being pulsed as a result of a WRTDC macro must have been previously assigned to the program (see SETDC macro).

DCWT
Direct Control Wait

GENERAL DESCRIPTION

◆ The DCWT macro is issued to wait a program after a WRTDC macro. A WRTDC macro is not considered complete until a DCWT macro is issued.

FORMAT

◆ The format of the DCWT macro is as follows:

Name	Operation	Operand
	DCWT	blank

SPECIFICATION RULES

Name Field

◆ Not required.

Operation Field

◆ DCWT.

Operand Field

◆ Blank.

Note

◆ 1. A WRTDC is considered complete when the Executive has issued the Write Direct instruction.

LOADP
Load-Program

GENERAL DESCRIPTION

◆ The LOADP macro permits a program to initiate another program internally without being terminated itself.

FORMAT

◆ The format of the LOADP macro is as follows:

Name	Operation	Operand
	LOADP	ptag

SPECIFICATION RULES

Name Field

◆ Not required.

Operation Field

◆ LOADP

Operand Field

◆ ptag is the address of an area whose first byte is a load status byte and whose succeeding bytes contain an E LOD message in the same format as required for the corresponding operator typein.

The LOADP SVC will WAIT the calling program until the load is completed or aborted. This is done to avoid the consequences of cases where a calling program is terminated before the load is successful. Special note should be taken by communications programs utilizing this macro, due to the possible timing implications. It is the responsibility of the calling program to interrogate the load status byte to determine the outcome of the LOADP request.

The format of the ptag area is:

X YYYY

where: X = load status byte

Hex

40 - Program load has been accomplished successfully.

01 - Program load is aborted because key in area is not free. Retry load again.

02 - Program load is aborted because of error.

yyy = E LOD message.

E LOD program,mn,p,pa,H,mmmmmm

See TDOS Operators Guide for explanation of format.

UPR
User Priority Relinquish

GENERAL DESCRIPTION

◆ The UPR macro permits a program to give up control to the program below it in the Operation List without waiting itself.

FORMAT

◆ The format of the UPR macro is as follows:

Name	Operation	Operand
	UPR	

SPECIFICATION RULES

Name Field

◆ Not required.

Operation Field

◆ UPR

Operand Field

◆ Blank

Note: Care should be taken when issuing this macro that there is a program below in the Operation List that is eligible for control or that there is an interrupt expected so that the system will not idle.

DCXT Exit From User's Direct Control Routine

GENERAL DESCRIPTION

◆ The DCXT macro is issued in the user direct control routine to return control to the user inline program after the Direct Control interrupt has been processed.

FORMAT

◆ The format of the DCXT macro is as follows:

Name	Operation	Operand
	DCXT	DCCB-name

SPECIFICATION RULES

Name Field

◆ Not required.

Operation Field

◆ DCXT.

Operand Field

◆ The *DCCB-name* is the symbolic name of the DCCB macro associated with the trunk to be handled. There must be a different DCCB macro for each trunk handled.

Note

◆ 1. The DCCB pointed to contains the in-line return address, BR 10 and BR 11, all of which are restored on a DCXT macro call.

Executive Communication Macro Expansion

Macro Name	Expansion Byte No.	Meaning
TYPE	0-1	Contains the Supervisor Call instruction generated for this macro. SVC Code = $(01)_{16}$
	2	Contains in the rightmost 7 bits a binary count for the length of the message to be typed. If a reply is requested, the high-order bit is set to one; otherwise it is zero.
	3-5	Contains the address of the message to be typed.
	6	Contains the binary count for the length of the reply if required.
	7-9	Contains the address of the area to receive the reply if required.
COMTY	0-1	Contains the Supervisor Call instruction generated for this macro. SVC Code = $(07)_{16}$
LPOV	0-1	Contains the Supervisor Call instruction generated for this macro. SVC Code = $(02)_{16}$
	2-7	Contains the 1-6 byte program load name as specified by the first operand of this macro.
	8-11	Contains the address to return to after the load as specified by the second operand of this macro.
TERM	0-1	Contains the Supervisor Call instruction generated for a QUIET macro. SVC Code = $(1C)_{16}$
	2-3	Contains the Supervisor Call instruction generated for this macro. SVC Code = $(09)_{16}$
TERMS	0-1	Contains the Supervisor Call instruction generated for a QUIET macro. SVC Code = $(1C)_{16}$
	2-3	Contains the Supervisor Call instruction generated for this macro. SVC Code = $(0A)_{16}$
	4-9	Contains the 1-6 byte name of the successor program.
TERMD	0-1	Contains the Supervisor Call instruction generated for this macro. SVC Code = $(16)_{16}$
STXIT	0-1	Contains the Supervisor Call instruction generated for this macro. SVC Code = $(04)_{16}$

Executive Communication Macro Expansion *(Cont'd)*

Macro Name	Expansion Byte No.	Meaning
STXIT (Cont'd)	2-5	Contains the address of the user's program check interrupt routine as specified by the second operand of this macro.
	6-9	Contains the address of the user's interval timer interrupt routine as specified by the third operand of this macro.
	10-13	Contains the address of the user's operator communication interrupt routine as specified by the fourth operand of this macro.
	14	Contains $(80)_{16}$ if parameter s is present and $(00)_{16}$ if it is omitted.
	15-17	Contains the address of the user's unrecoverable error interrupt routine as specified by the fifth operand of this macro.
ERXIT	0-1	Contains the Supervisor Call instruction generated for this macro: SVC Code = $(34)_{16}$
	2-5	Contains the address to which control is to be given, i.e., the first location of non-contingency coding to be executed.
EXIT	0-1	Contains the Supervisor Call instruction generated for this macro. SVC Code = $(12)_{16}$ for EXIT PR $(13)_{16}$ for EXIT TR $(14)_{16}$ for EXIT CR
ADEXT	0-1	Contains the Supervisor Call instruction generated for this macro. SVC Code = $(03)_{16}$
DMODE	0-1	Contains the Supervisor Call instruction generated for this macro. SVC Code = $(05)_{16}$
	2-7	Contains the symbolic unit name.
	8-11	Contains the address of a one-byte location in which the control byte for the 7-level tape specified in the first operand of this macro.
DTYPE	0-1	Contains the Supervisor Call instruction generated for this macro. SVC Code = $(06)_{16}$
	2-7	Contains the symbolic unit name.
	8-11	Contains the address of a one-byte location in which the device type code for the device specified in the first operand of this macro will be stored.

Executive Communication Macro Expansion *(Cont'd)*

Macro Name	Expansion Byte No.	Meaning
DDEV	0-1	Contains the Supervisor Call instruction generated for this macro. SVC Code = $(08)_{16}$
	2-7	Contains the name of the symbolic device to be deallocated.
ASCII	0-1	Contains the Supervisor Call instruction generated for this macro. SVC Code = $(10)_{16}$
EBCD	0-1	Contains the Supervisor Call instruction generated for this macro. SVC Code =$(11)_{16}$
GETOD	0-1	Contains the Supervisor Call instruction generated for this macro. SVC Code = $(17)_{16}$
	2-5	Contains the address of the first byte of a 6-byte area that will receive the time of day. Format of TOD will be HHMMSS. HH = hours MM = minutes SS - seconds Each byte will be an EBCDIC digit.
SETIC	0-1	Contains the Supervisor Call instructions generated for this macro. SVC Code = $(15)_{16}$
	2-6	Address of the first six bytes that contain a zoned decimal value in the form HHMMSS to be set into the user programs time clock.
GEPRT	0-1	Contains the Supervisor Call instruction generated for this macro. SVC Code = $(18)_{16}$
	2-6	Address of the first of six bytes that will receive the program time in the zoned decimal format HHMMSS.
CKPT	0-3	Load general register 15 with Address of IFCP.
	4-7	Branch to the executive call instruction.
	8-11	Address of IFCP.
	12-13	CNOP 2,4

Executive Communication Macro Expansion *(Cont'd)*

Macro Name	Expansion Byte No.	Meaning
CKPT (Cont'd)	14-15	Contains the Executive call instruction generated for this macro. SVC Code = $(0E)_{16}$
	16-21	VKKPED
	22-26	Contains the checkpoint identification specified in the third operand of this macro.
	27-29	Contains the restart address specified in the first operand of this macro.
	30-31	Contains the assignment halfword for the device on which the checkpoint records are written. This information must be supplied by the program. It is obtained from the CCB (bytes 14-15) generated for the checkpoint device.
	32-35	Contains the disc address pointer.
	36-39	Contains the address of the first DTF expansion as specified in the fourth operand of this macro.
	40	Contains the checkpoint error byte. If an unrecoverable error occurs while a checkpoint is in progress, the error byte indicates the error as follows:
		$(01)_{16}$ - An error has occurred in an operand specified in the checkpoint macro.
		$(02)_{16}$ - A 7-channel tape has been specified as the device on which checkpoint records are to be written and the pack/unpack mode is not on.
		$(10)_{16}$ - An unrecoverable error occurred during the last I/O operation executed to the checkpoint device.
		$(80)_{16}$ - An End of Tape (ET) condition has occurred while writting the checkpoint records.
		$(81)_{16}$ - An unrecoverable error has occurred during CKPT.
		$(82)_{16}$ - Service request not honored.
	41	Unused.
	42-45	Contains a Branch instruction to the program's error routine if any of the conditions described under the error byte (byte 26) occur during checkpoint. The error routine address is specified in the second operand of this macro. Note: If the error byte is zeros $(00)_{16}$ after checkpoint has been completed, control is transferred to the instruction following the checkpoint macro.

Executive Communication Macro Expansion *(Cont'd)*

Macro Name	Expansion Byte No.	Meaning
CKPT (Cont'd)	46-47	SVC 25 This Supervisor call causes an FCP Load (FLOAD macro) to be effected.
	48-53	"WFRTRT" the name of the routine to be FLOADed.
	54-57	Load address of WFRTRT.
	58-61	Return address of WFRTRT.
ASSGN	0-1	Contains the Supervisor Call instruction generated for this macro. SVC Code = $(ID)_{16}$
	2-5	Contains the address of the CCB whose symbolic device name is to be assigned.
SMODE	0-1	Contains the Supervisor Call instruction generated for this macro. SVC Code = $(1A)_{16}$
	2-3	Contains the assignment halfword provided by the user.
	4	Not used.
	5	Contains the write control mode to be established. This byte and the Write Control Code are exchanged. By issuing the SMODE a user can alternate modes without supplying additional information.
QUIET	0-1	Contains the Supervisor Call instruction generated for this macro. SVC Code = $(1C)_{16}$
TOCOM	0-1	CNOP 2,4
	2-3	Contains the Executive Call instruction generated for this macro. SVC Code = $(20)_{16}$
	4-7	Contains the address of the first byte in the user's program to be moved to the Common Data Area.
	8-11	Contains the number of bytes to be moved.
	12-15	Contains the value indicating the location relative to the beginning of the Common Data Area into which the first byte of data is to be moved.

Executive Communication Macro Expansion *(Cont'd)*

Macro Name	Expansion Byte No.	Meaning
EXCOM	0-1	CNOP 2,4
	2-3	Contains the Executive call instruction generated for the macro: SVC Code = $(21)_{16}$
	4-7	Contains the address of the destination of the first byte to be retrieved.
	8-11	Contains the number of bytes to be moved.
	12-15	Contains the value indicating the location relative to the beginning of the Common Data Area from which the first byte of data is to be retrieved.
LPOVR	0-1	Contains the Supervisor Call instruction generated for this macro. SVC Code = $(2C)_{16}$
	2-7	Contains the segment load name as specified in the first operand of this macro.
	8-11	Contains the address to return before the segment is loaded as specified by the second operand of this macro.
FLOAT	0-1	Contains the Supervisor Call instruction generated for this macro. SVC Code = $(23)_{16}$
	2-7	Contains the segment load name as specified in the first operand of this macro.
	8-11	Contains the address to return after the segment is loaded as specified by the third operand of this macro.
	12-15	Contains the absolute core address of where this segment is loaded as specified by the second operand of this macro.
FLODR	0-1	Contains the Supervisor Call instruction generated for this macro. SVC Code = $(2D)_{16}$
	2-7	Contains the segment load name as specified in the first operand of this macro.
	8-11	Contains the address to return before the segment is loaded as specified by the third operand of this macro.
	12-15	Contains the absolute core address of where this segment is loaded as specified by the second operand of this macro.

Executive Communication Macro Expansion *(Cont'd)*

Macro Name	Expansion Byte No.	Meaning
WRTDC	0-1	Contains the Supervisor Call instruction generated for this macro. SVC Code = $(33)_{16}$
	3	Contains the one-byte self-defining value which specifies the DC trunks to be transmitted over as specified by the first operand of this macro.
	4	Contains the one-byte self-defining value which specifies the byte to be transmitted as specified by the second operand of this macro.
DCWT	0-1	Contains the Supervisor Call instruction generated for this macro. SVC Code = $(34)_{16}$
DCXT	0-1	Contains the Supervisor Call instruction generated for this macro. SVC Code = $(35)_{16}$
	2-5	Contains the address of the DCCB indicating the trunk to be handled as specified by the first operand of this macro.
LOADP	0-1	Contains the Supervisor Call instruction generated for this macro. SVC Code = $(1F)_{16}$
	2-5	Contains the address of the area containing the E LOD command.
UPR	0-1	Contains the Supervisor Call instruction generated for this macro. SVC Code = $(30)_{16}$

Executive Communication Macro Expansion *(Cont'd)*

Macro Name	Expansion Byte No.	Meaning
WTOV	0-1	Contains the Supervisor Call instruction generated for this macro. SVC Code = $(2E)_{16}$
	2-5	Contains the address to return after the segment is loaded as specified by the first operand of this macro.
STDXC	0-1	Contains the Supervisor Call instruction generated by this macro. SVC Code = $(2F)_{16}$
	2-5	Contains the address of the CCB used to reference the DXC as specified by the first operand of this macro.
DXCXT	0-1	Contains the Supervisor Call instruction generated for this macro. SVC Code = $(30)_{16}$
	2-5	Contains the address of the CCB used to reference the DXC as specified by the first operand of this macro.
SETDC	0-1	Contains the Supervisor Call instruction generated for this macro. SVC Code = $(31)_{16}$
	2-5	Contains the address of the DCCB indicating the trunk to be handled as specified by the first operand of this macro.
DCCB	0	Contains the I-field in the Read Direct instruction.
	1	Used to store the input byte. Initially set to $(01)_{16}$.
	2-3	Reserved. Causes dcv address to be word-aligned and one back-up byte, if required, is stored here.
	4-9	Contains the address of the Direct Control routine.
	8-11	Contains the address of the user's in-line return address.
	12-15	Contains the contents of General Register 10 on entrance to the user supplied routine.
	16-19	Contains the contents of General Register 11 on entrance to the user supplied routine.
RELDC	0-1	Contains the Supervisor Call instruction generated for this macro. SVC Code = $(32)_{16}$
	3	Contains the one-byte self-defining value which specifies the DC trunks to be released as specified by the first operand of this macro.

MACRO EXPANSIONS (Cont'd)

Core Requirement Summary Table *(Cont'd)*

Executive Communication Macros	Required Number of Bytes
DCCB (TDOS only)	20
RELDC (TDOS only)	4
WRTDC (TDOS only)	4
DCWT (TDOS only)	2
DCXT (TDOS only)	6
UPR (TDOS only)	2
LOADP (TDOS only)	6

MACRO EXPANSIONS

Core Requirement Summary Table

Executive Communication Macros	Required Number of Bytes
TYPE	10
COMTY	2
LPOV	12
TERMS	8
TERM	2
TERMD	2
STXIT	18
ERXIT (TDOS only)	6
EXIT	2
ADEXT	2
DMODE	12
DTYPE	12
DDEV	8
ASCII	2
EBCD	2
GETOD	6
SETIC	6
GEPRT	6
CKPT	48
ASSGN	6
SMODE	6
QUIET	2
TOCOM	16
EXCOM	16
LPOVR (TDOS only)	12
FLOAT (TDOS only)	16
FLODR (TDOS only)	16
WTOV (TDOS only)	6
STDXC (TDOS only)	6
DXCXT (TDOS only)	6
SETDC (TDOS only)	6