E. 13.041

SC24-5211-1
File No. S370/4300-30

Program Product

VSE/Advanced Functions Application Programming: Macro Reference

IBM

SC24-5211-1
File No. S370/4300-30

Program Product

VSE/Advanced Functions Application Programming: Macro Reference

Program Number 5746-XE8

**Version 1 Release 3
Modification Level 5**

Second Edition (June 1983)

This is a major revision of SC24-5211-0. This edition applies to Version 1, Release 3, Modification Level 5 of VSE/Advanced Functions, Program Number 5746-XE8, and to all subsequent releases until otherwise indicated in new editions or Technical Newsletters.

Changes are made periodically to the information herein; before using this publication in connection with the operation of IBM systems, consult the latest IBM System/370 and 4300 Processors Bibliography, GC20-0001, for the editions that are applicable and current.

PREFACE

This publication contains, for the experienced programmer, reference information about data management and system control macros. It lists all the macros in one alphabetic sequence. For the most part, restrictions and programming details have been omitted in order to provide rapid access to the information in this publication. If the information included herein is not sufficient for your purposes, refer to the books in the following publications list:

VSE/Advanced Functions Application Programming: Macro User's Guide, SC24-5210

VSE/Advanced Functions, Data Management Concepts, GC24-5209

OS/VS - DOS/VSE - VM/370 Assembler Language, GC33-4010

The macro notation used is explained in the front of the manual.

CONTENTS

FIGURES

SUMMARY OF AMENDMENTS

Version 1, Release 3, Modification Level 5

This edition documents minor changes and additions to the following macros:

- EXTRACT
- GETIME
- MAPSSID
- STXIT

Additional changes include APAR corrections and various editorial corrections.

Version 1, Release 3, Modification Level 0

For a complete overview of functions and computing services new with this release of VSE/Advanced Functions, refer to the publication Introduction to the VSE System.

Technical Newsletter SN33-9293 documents the changes resulting from the:

- Extension of the record size for DTFDI input files to 81 bytes
- New I/O error handling concept
- New parameter for the ATTACH macro

Additional changes include APAR corrections and minor technical corrections.

MACRO NOTATION

There are two different types of macros: data management (IOCS) and system control macros. The data management macros define the characteristics of a file and identify the I/O operation to be performed on the file. The system control macros enable you to make use of various functions performed by programs and routines of VSE/Advanced Functions.

All macros are written in assembler format statements, that is, they consist of a number of fields, such as name field, operation field, and operand field.

Macro Fields

Macros, like assembler statements, have a name field, operation field, and operand field. Comments can also be included as in assembler statements, although certain macros require a comment to be preceded by a comma if the macro is issued without an operand. These macros are: CANCEL, DETACH, FREEVIS, GETIME, GETVIS, TESTT, and TTIMER.

The name field in a macro may contain a symbolic name. Some macros (for example, CCB, TECB, or DTFxx) require a name.

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

The operands in the operand field must be written in either positional, keyword, or mixed format.

Positional Operands

In this format, the operand values must be in the exact order shown in the individual macro discussion. Each operand, except the last, must be followed by a comma, and no embedded blanks are allowed. If an operand is to be omitted in the macro, and following operands are included, a comma must be inserted to indicate the omission. No commas need to be included after the last operand. Column 72 must contain a continuation punch (any non-blank character) if the operands fill the operand field and overflow onto another line.

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

```
  GET CDFILE,WORK
```

Keyword Operands

An operand written in keyword format can have this form:

```
LABADDR=MYLABELS
```

where LABADDR is the keyword, MYLABELS is a name you specify, and LABADDR=MYLABELS is the complete operand.

The keyword operands in the macro may appear in any order, and those that are not required may be omitted. Different keyword operands may be written in the same statement, each followed by a comma, except for the last operand of the macro.

Mixed Format

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

For more detailed information on coding macro statements, see OS/VS-DOS/VSE-VM/370 Assembler Language.

Notational Conventions

The following conventions are used in this book to illustrate the format of macros:

- Uppercase letters and punctuation marks (except as described in these conventions) represent information that must be coded exactly as shown.
- Lowercase letters and terms represent information which you must supply. More specifically, an n indicates a number, an r indicates a decimal register number, and an x indicates an alphameric character.
- Information contained within brackets [ ] represents an optional value that can be included or omitted, depending on the requirements of the program.
- Options contained within brackets and separated by an OR symbol (|) represent alternatives, one of which may be chosen. For example:

  ```
  [,entrypoint|,(0)]
  ```

- An underlined option represents the assumed default value in case you omit the operand. For example:

  [<u>A</u>|B|C]

- Options contained within braces { } and separated by an OR symbol (|) represent alternatives, one of which must be chosen. For example:

  {phasename|(1)}

- An ellipsis (a series of three periods) indicates that a variable number of items may be included.

Register Notation

Certain operands can be specified in either of two ways:

- You may specify the operand directly which results in code that, for example, cannot be executed in the SVA because it is not reentrant.

- You may load the address of the value into a register before issuing the macro. This way the generated code is reentrant and may be executed in the SVA. When using register notation, the register should contain only the specific address, and high-order bits should be set to 0.

When the macro is assembled, instructions are generated to pass the information contained in the specified register to IOCS or to the supervisor. For example, if an operand is written as (8), IOCS or the supervisor expects information to be stored at the address contained in general register 8. This is an example of ordinary register notation.

You can save both storage and execution time by using what is known as special register notation. In this method, the operand is shown in the format description of the macro as either (0) or (1), for example. This notation is special because the use of registers 0 and 1 is allowed only for the indicated purpose.

If special register notation is indicated by (0) or (1) in a macro format description and you use ordinary register notation, the macro expansion will contain an extra LR instruction, for example, LR 0,8.

The format description for each macro shows whether special register notation can be used, and for which operands. The following example indicates that the filename operand can be written as (1) and the workname operand as (0):

```
GET {filename|(1)}[,workname|,(0)]
```

If either of these special register notations is used, your program must load the designated register before executing the macro expansion. Ordinary register notation can also be used.

Operands in (S,address) Notation

Certain system control macros (for instance, ATTACH, GENIORB, GENL, LOAD) allow three notations for an operand:

- Register notation, as described in the preceding paragraph.
- Notation as a relocatable expression which, in the macro expansion, results in an A-type address constant.
- Notation in the form (S,address) which, in the macro expansion, results in the generation of an explicit address, that is, an assembler instruction address in base-displacement form. Address can be specified either as a relocatable expression, for example: (S,RELOC), or as two absolute expressions, the first of which represents the displacement and the second the base register, for example: (S,512(12)).

Consider using this notation if your program is to be reenterable. In a reenterable program, macro operands often refer to fields in dynamic storage. The (S,address) format offers an alternative to register notation: if two or more of such operands have to be provided for one macro, there is no need for loading addresses into that many registers.

MACRO DESCRIPTION

ASPL

Name	Operation	Operand
[name]	ASPL	[DSECT={<u>NO</u>|YES}]

The ASPL macro generates the parameter list (of 5 bytes length) which is used to pass information to the ASSIGN macro. For the format of the parameter list, see <u>VSE/Advanced Functions Application Programming: Macro User's Guide</u>.

DSECT={NO|YES}: Specify DSECT=YES if you want the parameter list to be generated as a mapping DSECT. If the operand is omitted, in-line code is generated.

ASSIGN

```
--------------------------------------------------
Name        Operation    Operand
--------------------------------------------------
[name]      ASSIGN       ASPL={name1|(r1)}
                         ,SAVE={name2|(r2)}
--------------------------------------------------
```

The ASSIGN macro is used to assign and unassign tape, disk, and unit-record devices dynamically. The system will select a free unit and assign it temporarily to any free programmer logical unit. Upon completion of the assignment, the logical and physical unit numbers to which the unit has been assigned are returned to the user. This information can be used by the RELEASE macro to release a unit dynamically when it is no longer needed.

A skeleton example that shows how tape drives are assigned and unassigned dynamically is given in VSE/Advanced Functions Application Programming: Macro User's Guide.

ASPL={name1|(r1)}: Specifies the address of the parameter list, in which you indicate the function (assign or unassign) to be performed. Use the mapping DSECT generated by the ASPL macro to interpret the fields in the parameter list.

SAVE={name2|(r2)}: Specifies a 72-byte save area that has to be reserved by the problem program.

Return Codes in Register 15

00	Assignment successful.
04	No free LUB entry found.
08	Device not found in PUB table.
0C	cuu has wrong device type.
10	cuu is down.
18	No free tape unit found.
1C	Invalid logical unit for unassign.
20	cuu reserved by space management or by pending mount request.
24	Invalid function code.
28	No GETVIS space available.
2C	Device to be unassigned is not assigned.
30	Device is owned by another partition.
34	Conflicting I/O assignment. Device is not assigned.

ATTACH

```
------------------------------------------------------------------
Name          Operation     Operand
------------------------------------------------------------------
[name]        ATTACH        {entrypoint|(S,entrypoint)|(r1)}
                            ,SAVE={savearea|(S,savearea)|(r2)}
                            [,ABSAVE={absavearea|(S,absavearea)|(r3)}]
                            [,ECB={ecbname|(S,ecbname)|(r4)}]
                            [,MFG={area|(S,area)|(r6)}]
                            [,RETURN={NO|YES}]
------------------------------------------------------------------
```

A subtask can be initiated by any other task of the partition with the ATTACH macro.

The maximum possible number of subtasks that can be initiated in the system at a time is determined by the NTASKS parameter of the SUPVR generation macro. The maximum number that can be specified is 208. Up to 31 subtasks can run concurrently within a partition, provided the overall limitation of NTASKS is not exceeded.

If the maximum number of subtasks is already attached, any attempt to attach another subtask will be unsuccessful. This is indicated to the attaching task by a 1 in high-order bit 0 in register 1. Register 1 then points to an unposted ECB in the supervisor that contains the reason code in byte 3. If byte 3 is zero, the maximum number of subtasks in the system (NTASKS value) is already attached. A non-zero value indicates that the maximum number of 31 subtasks is already running in the partition. The attaching task may use this ECB to enter a wait state. The ECB will be posted by the system whenever a task is available for attaching.

If the ATTACH macro successfully initiates a subtask, the attached task is given the lowest subtask priority. Register 1 of the attached task contains the address of the attaching task's save area; the other registers contain the same values as those of the attaching task at the time when the ATTACH was issued. The address in register 1 can be used as the second operand of a POST macro later in the job if task-to-task communication is desired.

Upon return from a successful ATTACH, register 0 of the attaching task contains the address of the byte immediately following the save area of the attached task.

If register notation is used in any of the macro operands, register 0 and 1 should not be specified.

entrypoint|(S,entrypoint)|(r1): The operand specifies the entrypoint of the subtask.

SAVE={savearea|(S,savearea)|(r2)}: The operand must provide the address of the save area for the subtask. The save area is 128 bytes in length (=16 double words).

If an interrupt occurs while the subtask is in control, the system saves in this save area the subtask's interrupt status information, general purpose registers, and the floating-point registers (see Figure 1).

Before issuing the ATTACH macro, move the subtask name in the first eight bytes of the save area. The name is used to identify the subtask in the event of a possible abnormal termination condition.

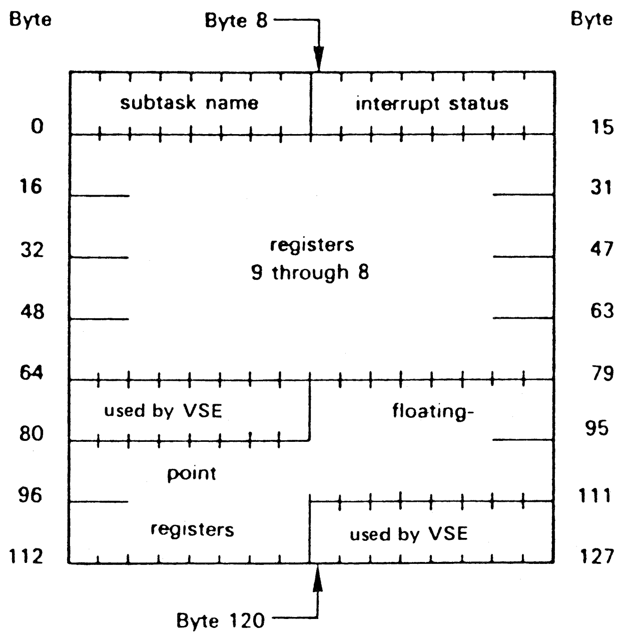


Figure 1. Subtask Save Area (128 bytes)

ABSAVE={absavearea|(S,absavearea)|(r3)}: Specify this operand only if the subtask is to use the attaching task's abnormal termination routine (see STXIT macro), that is, if it does not provide an abnormal termination routine of its own. The value specified in this operand must be the address of a 72-byte (doubleword-aligned) STXIT save area for the subtask. When an abnormal termination occurs, the supervisor saves the old PSW and general registers 0 through 15 in this area before the exit is taken.

ECB={ecbname|(S,ecbname)|(r4)}: The operand must be specified if other tasks can be affected by this subtask's termination, or if the ENQ and DEQ macro are used within the subtask. This parameter is the name of the subtask's event control block (ECB). The ECB may be any 4-byte field where in byte 2, bit 0 is the termination indicator and bit 1 is the abnormal indicator. The remaining bits of the four bytes are reserved. At the time a subtask is attached, bits 0 and 1 of byte 2 are set to 0. When a subtask terminates, the supervisor sets byte 2, bit 0 of the ECB to 1. In addition, byte 2, bit 1 is set to 1 when the subtask terminates abnormally; that is, if task termination is not the result of issuing the CANCEL, DETACH, RETURN, DUMP, JDUMP, or EOJ macros.

MFG={area|(S,area)|(r5)}: The operand is required if the program which issues the ATTACH macro is to be reenterable. It specifies the address of a 64-byte dynamic storage area, that is, storage which your program obtained through a GETVIS macro. This area is required for system use during execution of the macro.

RETURN={NO|YES}: Specify RETURN=YES if you want to activate a subtask for processing a subroutine or program that is terminated with a RETURN macro. If RETURN=YES is specified, the ATTACH macro generates a DETACH macro that will terminate the subtask after the RETURN macro has been encountered. Registers 14 and 15 are set up according to standard linkage conventions, that is, register 15 contains the address of the entry point of the routine specified in the ATTACH macro, and register 14 contains the return address (address of the DETACH macro generated by the ATTACH macro).

> **Note:** If your program uses VSAM files, you should provide a STXIT AB and PC macro and issue a CLOSE or TCLOSE for the files before canceling the subtask.

```
--------------------------------------------------
Name        Operation    Operand
--------------------------------------------------
[name]      CALL         {entrypoint|(15)}
                         [,(parameterlist)]
--------------------------------------------------
```

The CALL macro passes control from one program to a specified entry point in another program.

entrypoint|(15): Specifies the entry point to which control is passed. If the symbolic name of an entry point is specified, an instruction

```
  L 15,=V(entrypoint)
```

is generated as part of the macro expansion. The linkage editor makes the called program part of the calling program phase. The symbolic name must be either the name of a control section (CSECT) or an assembler language ENTRY statement operand in the called program. Control is given to the called program at this address. The called program resides in storage throughout execution of the calling program. This wastes storage if the called program is not needed throughout execution of the calling program.

If register 15 is specified, the entrypoint address must have been loaded into that register. Control is given to the called program at the address in register 15. Specifying register 15 preceded by a LOAD macro is most useful when the same program is called many times during execution of the calling program, but is not needed in storage throughout execution of the calling program.

parameterlist: Specifies one or more addresses (relocatable or absolute expressions) to be passed to the called program. Terms in the address must not be indexed. The addresses must be written in a sublist, with each address separated from the next by a comma. As part of the macro expansion, a parameter list is generated. It consists of a fullword for each address. Each fullword is aligned on a fullword boundary and contains the address to be passed in its three low-order bytes. The high-order bit in the last fullword is set to 1. When the called program is entered, register 1 (the parameter list register) contains the address of the parameter list.

CANCEL

Name	Operation	Operand
[name]	CANCEL	[ALL]

Issuing the CANCEL macro in a subtask abnormally terminates the subtask without branching to any abnormal termination routine. A CANCEL ALL macro issued in a subtask, or a CANCEL issued in the main task, abnormally terminates all processing in the partition (job). Job termination in multitasking causes all abnormal termination exits (via STXIT AB) to be taken for each task except the one that issued the CANCEL macro. Once these exits are taken, the job is terminated. Upon task termination, system messages (using the first eight bytes of each subtask save area) are issued to identify each subtask terminated.

If the CANCEL macro is issued without an operand, the macro must not contain a comment unless the comment begins with a comma. If CANCEL ALL is issued, the macro may include a comment.

If the DUMP option was specified, and SYSLST is assigned, a system dump will occur

- if a CANCEL ALL macro is issued by a subtask, or
- if a CANCEL macro is issued by a main task with subtasks attached.

Note: If your program uses VSAM files, ensure that these files are closed before you issue this macro.

CCB

```
-----------------------------------------------------
Name         Operation     Operand
-----------------------------------------------------
blockname    CCB           SYSnnn,command-list-name
                           [,X'nnnn'][,senseaddress]
-----------------------------------------------------
```

A CCB (command control block) macro must be specified in your program for each I/O device controlled by physical IOCS macros. The CCB (see Figure 2 on page 13) is necessary to communicate information to physical IOCS so that it can perform desired operations (for example, indicating printer channel 9). The CCB also receives status information after an operation and makes this available to your program. You should ensure proper boundary alignment of the CCB if this is necessary for your program.

Note: In some applications, it may be preferable to use an IORB (I/O Request Block) in place of a CCB. Do this by specifying either an IORB or GENIORB macro.

blockname: The CCB macro must be given a symbolic name (blockname). This name can be used as the operand in the EXCP and WAIT macros which refer to the CCB.

SYSnnn: This operand specifies the symbolic unit for the actual I/O unit with which this CCB is associated. The actual I/O unit can be assigned to the symbolic unit by an ASSGN job control statement.

command-list-name: This operand specifies the symbolic name of the first CCW used with a CCB. This name must be the same as the name specified in the assembler CCW statement that constructs the CCW.

X'nnnn': A hexadecimal value used to set the CCB user option bits. Column 5 of Figure 3 on page 16 gives the value used to set a user option bit 'on'. If more than one bit must be set, the sum of the values is used.

senseaddress: This operand, when supplied, indicates user error recovery (see Figure 3 on page 16, byte 2, bit 7) and generates a CCW for reading sense information (as the last field of the CCB). The name field (sense address) of the area that you supply must have a length attribute assigned of at least one byte. Physical IOCS uses this length attribute in the CCW to determine the number of bytes of sense information you desire.

CCB Format

From the above specifications, the macro sets up an area of either 16 bytes or 24 bytes. For the layout of this area and its contents see Figure 2 on page 13 and the following description of the individual fields.

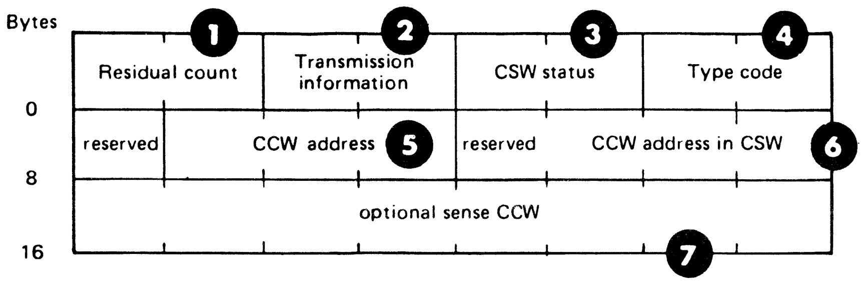


Figure 2. Layout and Contents of the Command Control Block (CCB)

1. After a record has been transferred, IOCS places the residual count from the CSW into bytes 0 and 1. By subtracting the residual count from the original count in the CCW, your program can determine the length of the transferred record. The field is set to zero for negative values.

2. Bytes 2 and 3 are used for transmission of information between physical IOCS and your program. For detailed information on the use and purpose of the individual bits in this field, see Figure 3 on page 16 . Your program can test any of the bits in this field using the mask given in the last column of Figure 3 on page 16 . Your program may test more than one bit by the hexadecimal sum of the test values.

   All bits are set to 0 when your program is assembled unless the X'nnnn' operand is specified. If this operand is specified, it is assembled into these two bytes. When your program is being executed, each bit may be set to 1 by your program (to request certain functions or specific feedback information) or by physical IOCS (as a result of having detected a particular condition). Any bits that can be turned on by physical IOCS during program execution are reset to zero by PIOCS the next time an EXCP macro is executed against the same CCB.

3. Bytes 4 and 5 are set to X'00' whenever an EXCP macro is issued against the CCB.

   The meaning of the bits in these two bytes is as follows:

Byte 4:	Byte 5:
0 = attention	0 = program-controlled interruption
1 = status modifier	1 = incorrect length
2 = control unit end	2 = program check
3 = busy	3 = protection check
4 = channel end	4 = channel data check
5 = device end	5 = channel control check
6 = unit check	6 = interface control check
7 = unit exception	7 = chaining check

If bit 5 of CCB byte 2 is set to 1 and device end results as a separate interrupt, device end will be posted.

4. Contents of byte 6:

   X'0u' = original CCB
   X'4u' = BTAM-ES CCB
   X'8u' = user-translated CCB in virtual partition

   Note:
   If u = 0: the address in byte 7 refers to a system logical unit.
   If u = 1: the address in byte 7 refers to a programmer logical unit

   Contents of byte 7:
   Hexadecimal representation of SYSnnn as follows:

SYSRDR = 00	SYS000 = 00
SYSIPT = 01	SYS001 = 01
SYSPCH = 02	SYS002 = 02
SYSLST = 03	.
SYSLOG = 04	.
SYSLNK = 05	.
SYSRES = 06	SYS254 = FE
SYSSLB = 07	
SYSRLB = 08	
SYSUSE = 09	
SYSREC = 0A	
SYSCLB = 0B	
SYSDMP = 0C	
SYSCAT = 0D	

5. Bytes 9-11 contain the address of the CCW (or of the first of a chain of CCWs) associated with the CCB:

   This is a real address if CCB byte 6 = X'8u'.
   This is a virtual address if CCB byte 6 = X'0u'.

6. Bytes 13-15 contain either of the following:

The CCW address contained in the CSW at channel-end interrupt for the I/O operation involving the CCB; or the address of the associated channel appendage routine if CCB byte 12 contains X'40'.

7. Bytes 16 to 23 are provided only if the sense operand was specified in the CCB macro. They accommodate the CCW for returning sense information to your program.

Byte	Bit	Condition Indicated: 1 (ON)	Condition Indicated: 0 (OFF)	ON Values for Third Operand in CCB Macro	Mask for Test Under Mask Instr.
2	0 Traffic Bit (WAIT).	I/O Completed. Normally set at Channel End. Set at Device End if bit 5 is on.	I/O requested and not completed.		X'80'
	1 End of File on System Input. 3211 UCB Parity Check (line complete).[8]	/* or /& on SYSRDR or SYSIPT. Byte 4, unit exception bit is also on. Yes			X'40'
	2 Unrecoverable I/O Error.	I/O error passed back due to program option or operator option.	No program or operator option error was passed back.		X'20'
	3 Accept Unrecoverable I/O Error (bit 2 is ON).[1]	Return to user after physical IOCS attempts to correct I/O error.[2]	Operator option: dependent on the error.	X'1000'	X'10'
	4 Return: DASD data checks, 3540 data checks, 2671 data checks, 1017/1018 data checks. 5424/5425 not ready. Indicate action type messages for DOC.[1]	Operator options: Ignore, Retry, or Cancel. Ignore or Cancel. Return to user.	Operator options: Retry or Cancel. Cancel.	X'0800'	X'08'

Figure 3 (Part 1 of 5). Conditions Indicated by CCB Bytes 2 and 3

Byte	Bit	Condition Indicated: 1 (ON)	Condition Indicated: 0 (OFF)	ON Values for Third Operand in CCB Macro	Mask for Test Under Mask Instr.
2	5 Post at device end. Specify this bit to be set on for a 2560 or 5424/5425.[1]	Device end condition is posted: that is, byte 2, bit 0 and byte 3, bits 2 and 6 set at device end. Also byte 4, bit 5 is set.	Device end conditions are not posted. Traffic bit is set at channel end.	X'0400'	X'04'
	6 Return: Uncorrectable tape read data check (2400 series or 3420); 1018, 2560 data check, 2520 or 2540 punch equipment check; 2560, 5424/5425 read, punch, print data, and print clutch equipment checks; 3881 equipm.check 3504, 3505, or 3525 perm. errors DASD read or read verify data check 3211 passback requested;[8] 3895 error codes requested[9] (data checks on count not retained).[1]	Return to user; after physical IOCS attempts to correct 3211[8], tape or DASD error; when 1018 or 2560 data check[4]; when 2560 or 5424 /5425 equipment check; when 3504, 3505, 3525, permanent error (byte 3, bit 3 is also on).	Operator option: Ignore or Cancel for tapes, paper tape punch(1018), card punches other than 2560 and 5424/5425. Retry or Cancel for DASD, 2560, or 5424/5425.	X'0200'	X'02'
	7 User Error Routine.[1]	User handles error recovery.[3]	A physical IOCS error routine is used unless the CCB sense address operand is specified. The latter requires user error recovery.	X'0100'	X'01'

Figure 3 (Part 2 of 5). Conditions Indicated by CCB Bytes 2 and 3

Byte	Bit	Condition Indicated 1 (ON)	Condition Indicated 0 (OFF)	ON Values for Third Operand in CCB Macro	Mask for Test Under Mask Instr.
3	0 Data check in DASD count field. Permanent error for 3330, 3340, 3350, 3375.	Yes-Byte 3, bit 3 is off; Byte 2, bit 2 is on.	No		X'80'
	Data check - 1287 or 1288.	Yes	No		
	MICR - SCU not operational.	Yes	No		
	3211 Print Check (equipment check).[8]	Yes	No		
	3540 special record transferred.[7]	Yes	No		
	1 DASD track overrun.	Yes	No		X'40'
	1017 broken tape.	Yes	No		
	Keyboard correction 1287 in Journal Tape Mode	Yes	No		
	3211 print quality error (equipment check).[8]	Yes	No		
	MICR intervention required.	Yes	No		
	2 End of DASD Cylinder.	Yes	No		X'20'
	Hopper Empty 1287/1288 Document Mode.	Yes	No		
	MICR - 1255/1259/1270/1275/1419 disengage.	Document feeding stopped.	No		
	1275/1419D, I/O error in external interrupt routine	Channel data check or Busout check.			
	3211/2245 line position error.[5 8]	Yes	No		

Figure 3 (Part 3 of 5). Conditions Indicated by CCB Bytes 2 and 3

Byte	Bit	Condition Indicated: 1 (ON)	Condition Indicated: 0 (OFF)	ON Values for Third Operand in CCB Macro	Mask for Test Under Mask Instr.
3	3 Tape read data check (2400 series); 2520, 2540 or 3881 equipment check; any DASD data check.	Operation was unsuccessful. Byte 2, bit 2 is also on. Byte 3, bit 0 is off.	No		X'10'
	1017, 1018 data check.	Yes	No		
	1287, 1288 equipment check.	Yes	No		
	2560, 3203, 5203, 5424/5425 read, punch, print data, and print clutch equipment checks.	Byte 2, bit 6 is also on.	No		
	3504, 3505, 3525 permanent errors.	Byte 2, bit 6 is also on.	No		
	3211 data check/ print check.[8]	Yes	No		
	3540 data check.	Yes	No		
	4 Nonrecovery Questionable Condition.	For card: unusual command sequence. For DASD, no record found. 1287, 1288 document jam or torn tape. 3211 UCB parity check (command retry). 5424/5425 not ready.			X'08'
	5 No-record-found condition (retry on DASD).[1]	Retry command if no-record-found condition occurs (disk).	Set the nonrecovery questionable condition bit on and return to user.	X'0004'	X'04'

Figure 3 (Part 4 of 5). Conditions Indicated by CCB Bytes 2 and 3

Byte	Bit	Condition Indicated 1 (ON)	Condition Indicated 0 (OFF)	ON Values for Third Operand in CCB Macro	Mask for Test Under Mask Instr.
3	6 Verify error for DASD or Carriage Channel 9 overflow.	Yes. (Set on when Channel 9 is reached. Only if Byte 2, bit 5 is on).	No		X'02'
	1287 document mode: late stacker select.	Yes	No		
	1288 End-of-Page (EOP).	Yes	No		
	7 Command Chain Retry. Specify this bit to be set on if command chaining is used.[1]	Retry begins at last CCW executed.[6] [10]	Retry begins at first CCW or channel program.	X'0001'	X'01'

Figure 3 (Part 5 of 5). Conditions Indicated by CCB Bytes 2 and 3

Notes:

1. User Option Bits. Set in CCB macro. Physical IOCS sets the other bits off at EXCP time and on when the specified condition occurs.

2. I/O program checks and I/O protection checks always terminate the program.

3. You may not handle Channel Control Checks and Interface Control Checks. The occurrence of a channel data check, unit check, or channel chaining check cause byte 2, bit 2 of the CCB to turn on, and completion of posting and dequeuing to occur. I/O program and protection checks always cause program termination. Incorrect length and unit exception are treated as normal conditions (posted with completion). Also, you must request device end posting (CCB byte 2, bit 5) in order to obtain errors after channel end.

4. Error correction feature for 1018 is not supported by physical IOCS. When a 1018 data check occurs and CCB byte 2, bit 6 is on, control returns directly to you with CCB byte 3, bit 3 turned on.

5. A line position error on the 3211 can occur as a result of an equipment check, data check, or FCB parity check.

6. If an error occurs, physical IOCS updates the CCW address in bytes 9 through 11 of the CCB that is used for the pertinent I/O operation. The original CCW address must therefore be restored before another I/O operation using the same CCB is issued.

7. A deleted or bad spot record has been read on a 3540 diskette. CCW chain broken, after CCW reads special record.

8. 3211 remarks apply also to 3211-compatible printers (that is, with device type code of PRT1).

9. 3895 error codes are returned in CCB byte 8. Refer to the 3895 Document Reader/Inscriber manuals for information on these error codes.

10. For tape error retry, the whole CCW chain is retried.

CDLOAD

```
--------------------------------------------------
Name        Operation    Operand
--------------------------------------------------
[name]      CDLOAD       {phasename|(1)}
                         [,PAGE={NO|YES}]
                         [,RETPNF={NO|YES}]
--------------------------------------------------
```

The CDLOAD macro loads the phase specified in the first parameter from a core image library into the partition GETVIS area. The phase is loaded only if it is not yet in either the partition GETVIS area or the SVA. CDLOAD returns control to the phase which issued the macro.

The CDLOAD macro must not be used for a phase that has been linked as a member of an overlay structure. Instead, use the LOAD macro without specifying a load address.

If a phase is to be loaded, CDLOAD determines the size of the phase, acquires the appropriate amount of GETVIS storage, and loads the phase into that storage.

After successfully loading the phase or if loading is not required (because the phase is already in the partition GETVIS area or in the SVA), registers contain values as follows:

register 0: the load address,

register 1: the entry point,

register 14: the length of the phase.

phasename|(1): For phasename, specify the name of the required phase. If register notation is used, the register must contain the address of an 8-byte field that holds the phase name as an alphameric character string.

PAGE={NO|YES}: If you want to have the phase loaded on a page boundary, specify PAGE=YES.

RETPNF={NO|YES}: Determines whether the issuing phase is canceled if the phase to be loaded does not exist in a core image library. With RETPNF=YES, the phase is not canceled; instead, control is returned to the issuing phase with the appropriate return code.

Return Codes in Register 15

After execution of the macro, register 15 contains one of the following return codes:

0	CDLOAD completed successfully.
4	The size of the (real) partition's GETVIS area is OK.
8	Not applicable.
12	Insufficient storage available in the GETVIS area.
16	The partition CDLOAD directory (also known as anchor table) is full.
20	The phase does not exist in a core image library (this return code occurs only with RETPNF=YES).
24	A move-mode phase was requested.
32	A hardware (storage) failure occurred in the requested real partition GETVIS area.

```
-----------------------------------------------------------
Name        Operation    Operand
-----------------------------------------------------------
[name]      CDMOD        [CONTROL=YES]
                         [,CRDERR=RETRY]
                         [,CTLCHR={ASA|YES}]
                         [,DEVICE=nnnn]
                         [,FUNC={R|P|I|RP|RW|RPW|PW}]
                         [,IOAREA2=YES]
                         [,RDONLY=YES]
                         [,RECFORM={FIXUNB|VARUNB|UNDEF}]
                         [,SEPASMB=YES]
                         [,TYPEFLE={INPUT|OUTPUT|CMBND}]
                         [,WORKA=YES]
-----------------------------------------------------------
```

The CDMOD macro defines a logic module for a card reader/punch file. If you do not provide a name for the module, IOCS generates a standard module name.

CONTROL=YES: Include this operand if the CNTRL macro is used with the module and its associated DTFs. The module also processes files for which the CNTRL macro is not used.

If this operand is specified, the CTLCHR operand must not be specified.

This operand cannot be specified if IOAREA2 is used for an input file or if an input file is used in association with a punch file (when the operand FUNC=RP or RPW is specified) on the 2560, 3525, or 5424/5425; in this case, however, this operand can be specified in the DTFCD and CDMOD for the associated punch file.

CRDERR=RETRY: Include this operand if error retry routines for the 2540 and 2520 punch-equipment check are included in the module. Whenever this operand is specified, any DTF used with the module must also specify the same operand. This operand does not apply to an input or a combined file.

CTLCHR={ASA|YES}: Include this operand if first-character stacker select control is used. ASA denotes the American National Standards character set, YES the System/370 character set (see Appendix A). Any DTF to be used with this module must have the same operand. If CTLCHR is included, CONTROL must not be specified. This operand does not apply to a combined file or to an input file.

DEVICE={2540|1442|2501|2520|2560P|2560S|2596|3504|3505|3525|5425P|5425S|3881}: Include this operand to specify the I/O device used by the module. The 'P' and 'S' included with the '2560' and '5425' parameters specify primary or secondary input hoppers;

regardless of which is specified, however, the module generated will handle DTFs specifying either hopper. Specify 5425P/S for 5424P/S. If you omit this operand, 2540 is the default.

Any DTF to be used with this module must have the same operand (except as just noted concerning the 'P' and 'S' specification for the 2560 or 5425).

FUNC={R|P|I|RP|RW|RPW|PW}: This operand specifies the type of file to be processed by the 2560, 3525, or 5424/5425. Any DTF used with the module must have the same operand. R indicates read, P indicates punch, and W indicates print.

When FUNC=I is specified, the file will be both punched and interpreted; no associated file is necessary to achieve this.

RP, RW, RPW, and PW specify associated files; when one of these parameters is specified for one file, it must also be specified for the associated file(s). Associated files can have only one I/O area each.

IOAREA2=YES: Include this operand if a second I/O area is used. Any DTF used with the module must also include the IOAREA2 operand. This operand is not required for combined files. This operand is not valid for associated files.

RDONLY=YES: This operand causes a read-only module to be generated. Whenever this operand is specified, any DTF used with the module must have the same operand.

RECFORM={FIXUNB|VARUNB|UNDEF}: This operand specifies the record format: fixed-length, variable-length, or undefined. Any DTF used with the module must have the same operand. If TYPEFLE=INPUT, TYPEFLE=CMBND, or FUNC=I, RECFORM must be FIXUNB. For the 3881, only RECFORM=FIXUNB is valid, which is also the default.

SEPASMB=YES: Include this operand only if the module is to be assembled separately. This causes a CATALR card with the module name (standard or user-specified) to be punched ahead of the object deck and defines the module name as an ENTRY point in the assembly. If the operand is omitted, the assembler assumes that the module is assembled together with the DTF and the problem program.

TYPEFLE={INPUT|OUTPUT|CMBND}: This operand generates a module for either an input, output, or combined file. Any DTF used with the module must have the same operand. For the 3881, only TYPEFLE=INPUT is valid, which is also the default.

WORKA=YES: This operand must be included if records are to be processed in work areas instead of in I/O areas. Any DTF used with the module must have the same operand. This operand is not valid for the 3881.

CDMOD

Standard CDMOD Names

Each name begins with a 3-character prefix (IJC) and continues with a 5-character field corresponding to the options permitted in the generation of the module.

CDMOD name = IJCabcde

a = F	RECFORM=FIXUNB (always for INPUT, CMBND, or FUNC=I files)
= V	RECFORM=VARUNB
= U	RECFORM=UNDEF

b = A	CTLCHR=ASA (not specified if CMBND)
= Y	CTLCHR=YES
= C	CONTROL=YES
= Z	CTLCHR or CONTROL not specified

c = B	RDONLY=YES and TYPEFLE=CMBND
= C	TYPEFLE=CMBND
= H	RDONLY=YES and TYPEFLE=INPUT
= I	TYPEFLE=INPUT
= N	RDONLY=YES and TYPEFLE=OUTPUT
= O	TYPEFLE=OUTPUT

d = Z	WORKA and IOAREA2 not specified
= W	WORKA=YES
= I	IOAREA2=YES
= B	WORKA and IOAREA2
= Z	WORKA=YES not specified (CMBND file only)

e = 0	DEVICE=2540, 3881
= 1	DEVICE=1442, 2596
= 2	DEVICE=2520
= 3	DEVICE=2501
= 4	DEVICE=2540 and CRDERR
= 5	DEVICE=2520 and CRDERR
= 6	DEVICE=3505 or 3504
= 7	DEVICE=3525 and FUNC=R/P or omitted
= 8	DEVICE=2560 and FUNC=R/P or omitted
= 9	DEVICE=5425 and FUNC=R/P or omitted
= A	DEVICE=3525 and FUNC=RP
= B	DEVICE=3525 and FUNC=RW
= C	DEVICE=3525 and FUNC=PW
= D	DEVICE=3525 and FUNC=I
= E	DEVICE=3525 and FUNC=RPW
= F	DEVICE=2560 and FUNC=RP
= G	DEVICE=2560 and FUNC=RW
= H	DEVICE=2560 and FUNC=PW
= I	DEVICE=2560 and FUNC=I
= J	DEVICE=2560 and FUNC=RPW
= K	DEVICE=5425 and FUNC=RP
= L	DEVICE=5425 and FUNC=RW
= M	DEVICE=5425 and FUNC=PW
= N	DEVICE=5425 and FUNC=I

= 0 DEVICE=5425 and FUNC=RPW

Subset/Superset CDMOD Names

All but one of the parameters are exclusive (that is, do not allow supersetting). A module name specifying C (CONTROL) in the **b** location is a superset of a module name specifying Z (no CONTROL or CTLCHR). A module with the name IJCFCIWO is a superset of a module with the name IJCFZIWO.

```
-------------------------------------------
              * * * * *
        I J C F A B B O
              V Y C I 1
              U + H W 2
                C I Z 3
                Z N   4
                  O   5
                      6
                      7
                      8
                      9
                      A
                      B
                      C
                      .
                      .
                      .
                      M
                      N
                      O
-------------------------------------------
   + Subsetting/supersetting permitted.
   * No subsetting/supersetting permitted.
-------------------------------------------
```

CHAP

Name	Operation	Operand
[name]	CHAP	

The CHAP macro lowers the priority of the issuing subtask. This issuing subtask now becomes the subtask with the lowest priority of all the subtasks within the partition.

A CHAP macro issued by the main task is ignored.

CHECK

```
-----------------------------------------------------
Name        Operation    Operand
-----------------------------------------------------
[name]      CHECK        {filename|(1)}
                         [,control-address|,(0)]
-----------------------------------------------------
```

The CHECK macro prevents processing until data transfer on an I/O operation is complete. It may be issued either after a READ or WRITE macro was issued for a work file, or after a READ was issued for a MICR file to ensure that data transfer is complete.

Because of differences in the way that IOCS posts CCB transmission information bits in the DTFs, you should always issue a CHECK macro to ensure that data transfer is complete before testing these bits. If the data transfer is completed without an error or other exceptional condition, CHECK returns control to the next sequential instruction. If an error condition is encountered, control is transferred to the ERROPT address. If ERROPT is not specified, processing continues at the next instruction. If end-of-file is encountered, control transfers to the EOFADDR address.

filename|(1): The operand specifies the name of the file associated with the record to be checked or, if register notation is used, the register containing a pointer to the field that contains this name. This name is the same as that specified for the DTFxx header entry for the file.

Issuing a CHECK macro after a READ on a MICR device allows you to query the MICR document buffer (see Figure 4 on page 31) and to specify the control-address operand.

control-address|(0): Indicates the address to which control passes when a buffer is waiting for data or when the file is closed. If register notation is used, the specified register must point to a field that contains this address.

The CHECK macro determines whether the MICR document buffer contains data ready for processing, is waiting for data, contains a special nondata status, or the file (filename) is closed. If the buffer has data ready for processing, control passes to the next sequential instruction. If the buffer is waiting for data, or the file is closed, control passes to the address specified for control address, if present. If the buffer contains a special nondata status, control passes to the ERROPT routine for you to examine the posted error conditions before determining your action. (See byte 0, bits 2, 3, and 4, of the document buffer). Return from the ERROPT routine to the next sequential instruction via a branch on register 14, or to the control address in register 0.

CHECK

If the buffer is waiting for data, or if the file is closed, and the control address is not present, control is given to you at your ERROPT address specified in the DTFMR macro.

If an error, a closed file, or a waiting condition occurs (with no control-address specified) and no ERROPT address is present, control is given to you at the next sequential instruction.

If the waiting condition occurred, byte 0, bit 5 of the buffer is set to 1. If the file was closed, byte 0, bits 5 and 6 of the buffer are set to 1.

Buffer Status Indicators		
Byte	Bit	Comment
0	0	The document is ready for processing (you need never test this bit).
	1	Unrecoverable stacker select error, but all document data is present. You may continue to issue GETs and READs.
	2	Unrecoverable I/O error. An operator I/O error message is issued. The file is inoperative and must be closed.
	3	Unit Exception. You requested disengage and all follow-up documents are processed. The LITE macro may be issued, and the next GET or READ engages the device for continued reading.
	4	Intervention required or disengage failure. This buffer contains no data. The next GET or READ continues normal processing. This indicator allows your program to give operator information necessary to select pockets for documents not properly selected and to determine unread documents.
	5	The program issued a READ, no document is ready for processing, byte 0, bits 0 to 2 are off, or the file is closed (byte 0, bit 6 is on). The CHECK macro interrogates this bit. Note: You must test bits 1 through 4 and take appropriate action. Any data from a buffer should not be processed if bits 2, 3, or 4 are on.
	6	The program has issued a GET or READ and the file is closed. Bit 5 also on.
	7	Reserved.
1	0	Your stacker selection routine turns this bit on to indicate that batch numbering update (1419 only) is to be performed in conjunction with the stacker selection for this document. The document is imprinted with the updated batch number unless a late stacker selection occurs (byte 3, bit 2).
	1-7	Reserved. Note: If bits 6 or 7 (byte 2) are on, bit 0 is ignored by the external interrupt routine. With the 1419 (dual address) only, batch numbering update cannot be performed with the stacker selection of auto-selected documents.

Figure 4 (Part 1 of 4). MICR Document Buffer Format

Buffer Status Indicators		
Byte	Bit	Comment
2*	0	For 1419 or 1275 (dual address) only. An auto-select condition occurred after the termination of a READ but before a stacker select command. The document is auto-selected into the reject pocket.
	1-3	Reserved.
	4	Data check occurred while reading. You should interrogate byte 3 to determine the error fields.
	5	Overrun occurred while reading. Byte 3 should be interrogated to determine the error fields. Overruns cause short length data fields. When the 1419 or 1275 is enabled for fixed-length data fields, bit 4 is set.
	6-7	The specific meanings of bits 6 and 7 depend on the device type, the model, and the Engineering Change level of the MICR reader; but if either bit is on, the document(s) concerned is (are) auto-selected into the reject pocket. 1. 1412 or 1270: Bit 6 on indicates that a late read condition occurred. Bit 7 on indicates that a document spacing error occurred. (Unique to the 1270: both the document and the previous document are auto-selected into the reject pocket when this bit is on. This previous document reject cannot be detected by IOCS, and byte 5 of its document buffer does not reflect that the reject pocket was selected). 2. 1275 and 1419 (single address) without engineering change 125358: Bit 6 indicates that either a late read condition or a document spacing error occurred. Bit 7 indicates a document spacing error for the current document. 3. 1255, 1259, 1275, and 1419 (single or dual address) with engineering change 125358: Bit 6 indicates that an auto-select condition occurred while reading a document. The bit is set at the termination of the READ command before the stacker select routine receives control. Bit 7 is always zero.
* Byte 2 (bits 4, 5, 6, and 7) contains MICR sense information.		

Figure 4 (Part 2 of 4). MICR Document Buffer Format

Buffer Status Indicators		
Byte	Bit	Comment
3*	0	Field 6 valid.**
	1	Field 7 valid.**
	2	A late stacker selection (unit check late stacker select on the stacker select command). The document is auto-selected into the reject pocket.
	3	Amount field valid (or field 1 valid).**
	4	Process control field valid (or field 2 valid).**
	5	Account number field valid (or field 3 valid).**
	6	Transit field valid (or field 4 valid).**
	7	Serial number field valid (or field 5 valid).**
		Notes: 1. For the 1270, bits 3-7 are set to zero when the fields are read without error. 2. For the 1255, 1259, 1275, and 1419, bits 3-7 are set on when each respective field, including bracket symbolds, is read without error. This applies to bits 0, 1, and 3-7 on the 1259 and 1419 model 32. 3. For the 1255, 1259, 1275, and 1419, unread fields contain zero bits. Errors are indicated when an overrun or data check condition occurs while reading the data field.
* Byte 3 contains MICR sense information. ** Only for the 1259 model 34 or 1419 model 32. Bits 0 and 1 are not used for other models.		
4		Inserted pocket code determination by your stacker select routine. Whenever byte 0, bits 2, 3, or 4 are on, this byte is X'00' because no document was read and your stacker selection routine was not entered. Whenever auto-selection occurs, this value is ignored. A no-op (X'03') is issued to the device, and a reject pocket value (X'CF') is placed in byte 5. The pocket codes are (byte 2, bit 6 or 7 on): Pocket A: X'AF'* — Pocket 5: X'5F' Pocket B: X'BF'** — Pocket 6: X'6F' Except 1270 Pocket 0: X'0F' — Pocket 7: X'7F' models 1 and 3 Pocket 1: X'1F' — Pocket 8: X'8F' Pocket 2: X'2F' — Pocket 9: X'9F' Pocket 3: X'3F' — Reject Pocket: X'CF' Pocket 4: X'4F'
* 1275, 1419, and 1270 models 2 and 4 only. ** 1275 and 1419 only.		

Figure 4 (Part 3 of 4). MICR Document Buffer Format

Buffer Status Indicators		
Byte	Bit	Comment
5		The actual pocket selected for the document. The contents are normally the same as that in byte 4. Note: X'CF' is inserted whenever auto-selection occurs (byte 2, bit 6; byte 2, bit 7; byte 2, bit 0; or byte 3, bit 2). These conditions may result from late READ commands, errant document spacing, or late stacker selection. a. Start I/O for stacker selection is unsuccessful (byte 0, bit 1). b. An I/O error occurs (for example, invalid pocket code) on the 1419 (dual address) secondary control unit when selecting this document.

Additional User Work Areas
This additional buffer area can be used as a work area and/or output area. Its size is determined by the DTFMR ADDAREA operand. The only size restriction is that this area, plus the 6-byte status indicators and data portion must not exceed 256 bytes. This area may be omitted.

Document Data Area
The document data area immediately follows your work area. The data is right-adjusted in the document data area. The length of this data area is determined by the DTFMR RECSIZE operand.

Figure 4 (Part 4 of 4). MICR Document Buffer Format

CHKPT

```
----------------------------------------------------------------
Name          Operation     Operand
----------------------------------------------------------------
[name]        CHKPT         SYSnnn,{restart-address|(r1)}
                            [,end-address|,(r2)]
                            [,tpointer|,(r3)]
                            [,dpointer|,(r4)]
                            [,filename|,(r5)]
----------------------------------------------------------------
```

The CHKPT macro is used to record the status of your program so that the program, should its execution be terminated before it has completed processing, may be restarted using job control. The partition in which the program is to be restarted must start at the same location as when the program was checkpointed, and its end address must not be lower than the end address at checkpoint time. If the CHKPT macro is successfully executed, control is returned with the checkpoint number in unpacked decimal format in register 0. If it is unsuccessful and the checkpoint has not been taken, register 0 contains zero and the reason is printed on SYSLOG.

> **Note:** If a program using routines in the SVA is being checkpointed, you must make sure that SVA routines occupy the same locations on restart, should a restart become necessary.

Special register notation cannot be used with any of the CHKPT macro operands.

All VSAM files must be closed before the CHKPT macro is issued. A SAM ESDS (supported by the VSE/VSAM Space Management for SAM feature) cannot be repositioned by the restart program.

SYSnnn: Specifies the logical unit on which the checkpoint information is to be stored. It must be an EBCDIC magnetic tape or a DASD volume.

restart-address|(r1): Specifies a symbolic name of the instruction (or register containing the address) at which execution is to restart if processing must be continued later.

end-address|(r2): A symbolic name assigned to (or register containing the address of) the uppermost byte of the program area required for restart. This address must be higher than the highest address of storage occupied by any phase loaded into the partition. The address should be a multiple of 2K. If the address is not a multiple of 2K, it is rounded to the next 2K boundary. If this operand is omitted, all storage allocated to the partition (other than the GETVIS area) is checkpointed.

The specified end address is ignored if any GETVIS request was executed in the partition. (Note that GETVIS storage may have been requested by included IBM routines). In this case again, all storage allocated to the partition is checkpointed.

tpointer|(r3): Address of an 8-byte field containing 2 V-type address constants used in repositioning magnetic tape at restart time. The address may be a symbolic address or contained in a register.

The first constant points to a table containing the file names of all logical IOCS magnetic tape files to be repositioned. Each file name points to the corresponding DTF table where IOCS maintains repositioning information.

The second constant points to a table containing repositioning information for physical IOCS magnetic tape files to be repositioned. The entries in the table are:

- First halfword: hexadecimal representation of the logical unit address of the tape (copy from CCB).
- Second halfword: number of files within the tape (in binary notation), that is, the number of tape marks between the beginning of the tape and the position at checkpoint.
- Third halfword: number (in binary notation) of physical records between the preceding tape mark and the position at checkpoint.

If the first, second, or both constants are zero, no tapes are repositioned.

If the tables are contained in the same source module as the CHKPT macro, the constants must be defined as A-type constants.

dpointer|(r4): Address of a DASD operator verification table, used to allow the operator to verify DASD volume serial numbers at restart time. May be a symbolic address or contained in a register.

The entries in the table must consist of the following two halfwords:

- The logical unit number (in hexadecimal notation) of each DASD unit used by your program (copied from CCB bytes 6 and 7).
- Reserved.

There must be one entry for each DASD unit to be verified. At restart time, the volume serial number of each of these DASD units is printed on SYSLOG.

filename|(r5): The name of the associated DTFPH; used only for checkpoint records on disk.

CLOSE, CLOSER

```
-----------------------------------------------------
Name         Operation     Operand
-----------------------------------------------------
[name]       CLOSE         {filename1|(r1)}
                           [,filename2|,(r2)]...
-----------------------------------------------------
```

The format of the CLOSER macro is the same as that of the CLOSE macro, except that you code CLOSER instead of CLOSE in the operation field.

The CLOSE or CLOSER macro is used to deactivate previously opened files; it ends the association between a logical file declared in a program and a specific physical file on an I/O device.

A file may generally be closed at any time, with the following exceptions:

- Console files must not be closed; the CLOSE(R) macro is invalid for files defined by means of the DTFCN.
- Files assigned to an FBA device may not be closed in an ERROPT routine.

Files (such as on an FBA device) that use control interval format must be closed in order to ensure that data in the control interval buffer is physically written on the FBA device.

No further commands can be issued to the closed file until it is reopened.

When CLOSER is specified, the symbolic address constants that CLOSER generates from the parameter list are self-relocating. When CLOSE is specified, the symbolic address constants are not self-relocating. Throughout the manual the term CLOSE also implies CLOSER, unless stated otherwise.

filename|(r1): Enter the symbolic name of the file (DTF filename) to be closed in the operand field. A maximum of 16 files may be closed with one macro by entering additional filenames. Alternatively, you can load the address of the filename in a register and specify the register using ordinary register notation. The high-order 8 bits of this register must be zeros. For CLOSER, the address of filename may be preloaded into any of the registers 2 through 15. For CLOSE, the address of filename may be preloaded into register 0 or any of the registers 2 through 15.

> **Note:** If CLOSE or CLOSER is issued to an unopened tape input file, the option specified in the DTF rewind option is performed. If CLOSE or CLOSER is issued to an unopened tape output file, no tapemark or labels are written.

CNTRL

```
--------------------------------------------------------------
Name         Operation    Operand
--------------------------------------------------------------
[name]       CNTRL        {filename|(1)},code[,n1][,n2]
--------------------------------------------------------------
```

The CNTRL (control) macro provides commands for magnetic tape units, card devices, printers, DASDs, and optical readers. Commands apply to physical nondata operations of a unit and are specific to the unit involved. They specify such functions as rewinding tape, card stacker selection, and line spacing on a printer. For optical readers, commands specify marking error lines, correcting a line for journal tapes, document stacker selecting, or ejecting and incrementing documents. The CNTRL macro does not wait for completion of the command before returning control to you, except when certain mnemonic codes are specified for optical readers.

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

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

The CNTRL macro is ignored if specified for DTFSD or DTFDI DASD files.

filename|(1): Must be the name of the file specified in the DTF header entry. It can be specified as a symbol or in register notation.

code: Is the mnemonic code for the command to be performed. This must be one of a set of pre-determined codes as shown in Figure 5 on page 39 .

n1: Is required whenever a number is needed for stacker selection, immediate printer carriage control, or for line or page marking on the 3886.

n2: Applies to delayed spacing or skipping or to timing mark check on the 3886. In the case of a printer file, the parameters n1 and n2 may be required. If n1 is omitted and n2 is specified, a comma must be coded for n1.

IBM Unit	Mnemonic Code	n_1	n_2	Command
2400, 3410, 3420, 8809 Magnetic Tape Units	REW			Rewind Tape
	RUN			Rewind and Unload Tape
	ERG			Erase Gap (Writes Blank Tape)
	WTM			Write Tapemark
	BSR			Backspace to Interrecord Gap
	BSF			Backspace to Tapemark
	BSL			Backspace Logical Record
	FSR			Forward Space to Interrecord Gap
	FSF			Forward Space to Tapemark
	FSL			Forward Space Logical Record
1442, 2520 Card Read Punch	SS	1 2		Select Stacker 1 or 2
	E			Eject to Stacker 1 (1442 only)
2540 Card Read Punch 3504, 3505 Card Readers 3525 Card Punch	PS	1 2 3		Select Stacker 1, 2, or 3 (For 3504, 3505, and 3525, 3 Defaults to Stacker 2)
2560 Multi-Function Card Machine	SS	1 2 3 4 5		Select Stacker 1, 2, 3, 4, or 5
2596 Card Read Punch	SS	1 2		Select Stacker 1 for Read, or Stacker 3 for Punch Select Stacker 2 for Read, or Stacker 4 for Punch
5424/5425 Multi-Function Card Unit	SS	1 2 3 4		Select Stacker 1, 2, 3, or 4
1403, 1443, 3203, PRT1, 3800, 5203 Printers 3525 Card Punch with Print Feature[1]		See Note		
	SP	c	d	Carriage Space 1, 2, or 3 lines
	SK	c	d	Skip to Channel c and/or d (For 3525, a Skip to Channel 1 is Valid Only for Print Only Files)
1403, 5203 Printers with Universal Character Set Feature or 3203, PRT1, or 3800 Printers[1]	UCS	ON OFF		Data Checks are Processed with an Operator Indication Date Checks are ignored and Blanks are Printed
PRT1 Printer[1]	FOLD			Print Upper Case Characters for any Byte with Equivalent Bits 2-7
	UNFOLD			Print Character Equivalents of any EBCDIC Byte
2311, 2314, 2319, 3330, 3333, 3340, 3344, 3350, 3375, DASD[2]	SEEK			Seek to Address
3881 Optical Mark Reader	PS	1 2		Select Stacker 1 or 2
1287 Optical Reader	MARK			Mark Error Line in Tape Mode
	READKB			Read 1287 Keyboard in Tape Mode
	EJD			Eject Document
	SSD	1 2 3 4		Select Stacker A, B, Reject, or Alternate Stacking Mode
	ESD	1-4		Eject Document and Select Stacker
	INC			Increment Document at Read Station
1288 Optical Page Reader	ESD	1 3		Select Stacker A Reject Stacker (R)
	INC			Increment Document at Read Station
3886 Optical Character Reader	DMK	name (r) number		Page mark the document when it is stacker selected as specified in parameter n_1.
	LMK	name (r) number, number		Line mark the document when it is stacker selected as specified in parameter n_1.
	ESP	1 2	name (r) number	Eject and stacker select the current document to stacker A or B. Perform line mark station timing mark check as indicated in parameter n_2.

c An Integer that Indicates Immediate Printer Control (before printing).

d An Integer that Indicates a Delayed Printer Control.

[1] Note: PRT1 refers to 3211-compatible printers (that is, with a device type of PRT1).

[2] Note: This includes the 3350 operating in 3330 compatibility mode.

Figure 5. CNTRL Macro Command Codes

COMRG

Name	Operation	Operand
[name]	COMRG	[REG=r]

The COMRG macro places the address of the communication region of the partition from which the macro is issued into the specified register. If the operand is omitted, register 1 is assumed.

CPCLOSE

Name	Operation	Operand
[name]	CPCLOSE	[arglist|(r1)]

In spooling programs written in Basic Assembler Language, the CPCLOSE macro can be used to issue a CPCLOSE command to VM/370 in order to release a print or punch file for output.

> **Note:** The CPCLOSE macro is valid only if the supervisor was generated with the VM=YES option specified on the SUPVR macro.

arglist|(r1): This operand specifies a 16-byte argument list whose format is described below and which must be set up before issuing the macro. If the argument list name is specified, the system loads the address into register 1. If a register is specified, it is assumed to contain the address of the argument list and this address is loaded into register 1. If no operand is specified, register 1 is assumed to contain the address of the argument list.

0	Hexadecimal device address	EBCDIC device address	Job name
0 2	4	8	15

Device address = unit record device address of the device to be closed.

Return Codes in Register 15

X'00' - Successful completion of CPCLOSE macro.

X'04' - Device is invalid, no CLOSE is issued.

X'08' - VM=YES support not included in the supervisor.

DEQ

Name	Operation	Operand
[name]	DEQ	{rcbname|(0)}

A task releases a resource by issuing the DEQ macro. If other tasks are enqueued on the same RCB, the DEQ macro frees from their wait condition all other tasks that were waiting for that resource. In such cases, the highest priority task either obtains or maintains control. A task that attempts to dequeue a resource that was not enqueued or that was enqueued by another task is abnormally terminated. Dequeuing under these two conditions within an abnormal termination routine results in a null operation instruction.

rcbname|(0): The operand is the same as that in the ENQ macro and specifies the address of the RCB.

DETACH

```
---------------------------------------------------
Name        Operation    Operand
---------------------------------------------------
[name]      DETACH       [SAVE={savearea|(1)}]
---------------------------------------------------
```

The DETACH macro terminates execution of a task. A subtask is normally terminated by issuing a DETACH macro in the main task or in the subtask itself.

The DETACH macro sets byte 2, bit 0 of the ECB to 1 (if specified in the ATTACH macro) to indicate task termination. All tasks waiting on this ECB are taken out of the wait state, and the highest priority task obtains control.

If the subtask issues a DETACH macro without an operand, only the subtask issuing the DETACH macro is terminated. Any subtasks attached by the terminating subtask are not affected by the termination.

If the main task issues the DETACH macro without specifying an operand, it will be canceled, that is, all processing in the partition is terminated abnormally.

SAVE={savearea|(1)}: A subtask may also terminate a subtask it attached by issuing the DETACH macro with the SAVE operand. If the main task issues the DETACH macro with the SAVE operand, it can terminate any subtask in the partition. The SAVE operand provides the address of the savearea specified in the ATTACH macro for the subtask to be terminated.

> **Note:** If the subtask being terminated uses VSAM files, ensure that these files are closed before you issue this macro.

```
-------------------------------------------------------
Name         Operation    Operand
-------------------------------------------------------
[name]       DFR          FONT=code
                          [,BCH=n]
                          [,BCHSER=n]
                          [,CHRSET=n]
                          [,EDCHAR=(x,...)]
                          [,ERASE={NO|YES}]
                          [,NATNHP={NO|YES}]
                          [,REJECT=x]
-------------------------------------------------------
```

The DFR macro defines attributes common to a group of line types on a 3886.

FONT=code: Specifies the default font for all fields described by the format record. The default font is used to read a field unless another font is specified for an individual field through the DLINT macro.

This is the only required operand in the DFR macro. The valid codes and the fonts they represent are:

NUMA	Numeric OCR-A font
ANA1	Alphameric OCR-A font (mode 1)
ANA2	Alphameric OCR-A font (mode 2)
NUMB	Numeric OCR-B font (mode 3)
ANB1	Alphameric OCR-B font
NHP1	Numeric hand printing (normal mode)
NHP2	Numeric hand printing (verify mode)
GOTH	Gothic font
MRKA	Mark OCR-A font
MRKB	Mark OCR-B font

For a description of these fonts, see the appropriate IBM 3886 device manuals.

BCH={1|2|3}: Indicates that batch numbering is to be performed by the 3886. Specifying 1, 2, or 3 indicates that documents routed to a stacker are to be batch numbered. Specifying 1 indicates stacker A, 2 indicates stacker B, 3 indicates both stackers. If this operand is specified, the BCHSER operand is invalid. If neither BCH nor BCHSER are entered, no batch numbering is performed. This operand is valid only if the serial numbering feature is installed on the 3886. For more information on batch numbering, see the appropriate IBM 3886 device manuals.

BCHSER={1|2|3}: Indicates that both batch and serial numbering are to be performed by the 3886. Specifying 1, 2, or 3 indicates that documents routed to a stacker are to be batch and serial numbered. Specifying 1 indicates stacker A, 2 indicates stacker B, 3 indicates both stackers. If this operand is specified, the BCH operand is invalid. If neither BCH nor BCHSER is specified, batch and serial numbering are not performed. This operand is valid only if the serial numbering feature is installed on the 3886. For more information on batch and serial numbering, see the appropriate IBM 3886 device manuals.

CHRSET={0|1|2|3|4|5}: Specifies which one of the options in Figure 6 is to be used for recognizing characters. If this operand is not entered, 0 is assumed.

OCR-A			OCR-B			
Numeric Mode	Alphameric Modes		Numeric Mode	Alphameric Mode		
Highspeed Printers or Typewriters	Mode 1 (Highspeed) Printer)	Mode 2 (Typewriter)	Highspeed Printers or Typewriters		Hexa-decimal Code	Format Record Codes
$	$	$	$	$	5B	00
£	£	£	£	£	5B	01
¥	¥	¥	¥	¥	5B	02
	N̄	N̄		Ñ	7B	
$	$	$	$	$	5B	03
	Å	Å		Å	5B	
	Æ	Æ		Æ	7B	
	Ø	Ø		Ø	7C	04
$	$	$		Ü Note	5B	
	Ä	Ä		Ä	7B	
	Ö	Ö		Ö	7C	
	Ü	Ü			F0	05

Note: In OCR-A font the Ü is coded as a zero and should be used only in alphabetic fields.

Figure 6. Character Set Option List

EDCHAR=(x,...): Specifies up to six characters that may be deleted from any field that is read. The EDCHAR parameter in the EDITn keyword of the DLINT macro controls this function for individual fields. If this operand is omitted, no character deletion is performed. See the note under the REJECT operand discussion for characters that must be specified in quotes. For example, to specify the characters &, >, and), you would code EDCHAR=('&','>',')').

ERASE={NO|YES}: Specifies whether group and character erase symbols are to be recognized as valid symbols. If this operand is not specified, NO is assumed. For more information on group and character erase symbols, see the appropriate IBM 3886 device manuals.

NATNHP={NO|YES}: Specifies which of the numeric hand printing character set options are used for the numbers 1 and 7. YES indicates that the European Numeric Hand Printing (ENHP) characters 1 and 7 are used; NO indicates the Numeric Hand Printing (NHP) characters 1 and 7 are used. If this operand is not entered, NO is assumed.

REJECT=x: Indicates the character that is to be substituted in the data record for any reject character read by the device. If this operand is omitted, X'3F' is assumed. Reject characters are characters that are not recognizable by the device.

> **Note:** This note applies to the keywords REJECT and EDCHAR. Apostrophes enclosing the character are optional for all characters except special characters used in macro operands. For a description of these characters, see the manual OS/VS-DOS/VSE-VM/370 Assembler Language .

DIMOD

```
----------------------------------------------------------
Name          Operation    Operand
----------------------------------------------------------
[name]        DIMOD        [IOAREA2=YES]
                           [,RDONLY=YES]
                           [,SEPASMB=YES]
                           [,TRC=YES]
                           [,TYPEFLE={OUTPUT|INPUT}]
----------------------------------------------------------
```

The DIMOD macro defines a logic module for a device-independent file. If you do not provide a name for the module, IOCS generates a standard module name.

For DASD devices, a user-supplied logic module is not required. If one is supplied, it is ignored. OPEN always provides linkage to an IBM-supplied logic module which resides in the SVA.

IOAREA2=YES: Include this operand if a second I/O area is needed. A module with this operand can be used with DTFDI specifying either one or two I/O areas. If the operand is omitted or is invalid, one I/O area is assumed.

RDONLY=YES: This operand causes a read-only module to be generated. Whenever this operand is specified, any DTF used with the module must have the same operand.

SEPASMB=YES: Include this operand only if the module is to be assembled separately. This causes a CATALR card with the module name (standard or user-specified) to be punched ahead of the object deck and the module name to be defined as an ENTRY point in the assembly. If the operand is omitted, the assembler assumes that the module is assembled together with the DTF and the problem program.

TRC=YES: Include this operand to specify whether the module is to test the table reference character indicator in the DTFDI or ignore that indicator. If TRC=YES is specified, the generated module can process output files with table reference characters and those without. If the TRC operand is specified, TYPEFLE=INPUT must not be specified.

TYPEFLE={OUTPUT|INPUT}: Include this operand to specify whether the module is to process input or output files. If OUTPUT is specified, the generated module can process both input and output files.

DIMOD

Standard DIMOD Names

Each name begins with a 3-character prefix (IJJ) followed by a 5-character field corresponding to the options permitted in the generation of the module.

DIMOD name = IJJabcde

a	= F	always
b	= C	RPS=SVA is not specified
	= V	RPS=SVA
c	= B	TYPEFLE=OUTPUT (both input and output)
	= I	TYPEFLE=INPUT
d	= I	IOAREA2=YES
	= Z	IOAREA2=YES is not specified
e	= C	RDONLY=YES
	= D	RDONLY=YES is not specified

Subset/Superset DIMOD Names

All of the operands except TRC=YES allow subsetting. A module name specifying B is a superset of the module specifying I, for example. IJJFCBID is a superset of the module IJJFCIID.

The IBM-supplied preassembled logic modules do not have TRC=YES. The system programmer can reassemble them with TRC=YES to support 3800 table reference characters. Although the code that is generated for a module assembled with TRC=YES is different from the code that is generated for a module with TRC=NO, the module name is the same. If some, but not all DIMOD logic modules are reassembled this way, it may interfere with subsetting or supersetting.

```
---------------------------------------------
            * + + *
    I J J F C B I C
            V I Z D
---------------------------------------------
   + Subsetting/supersetting permitted.
   * No subsetting/supersetting permitted.
---------------------------------------------
```

DISEN

```
--------------------------------------------------
Name        Operation    Operand
--------------------------------------------------
[name]      DISEN        {filename|(1)}
--------------------------------------------------
```

This macro stops the feeding of documents through the magnetic character reader or optical reader/sorter. The program proceeds to the next sequential instruction without waiting for the disengagement to complete. You should continue to issue GET or READ until the unit exception bit (byte 0, bit 3), of the buffer status indicators is set on (see Figure 4 on page 31).

filename|(1): Specifies the name of the file to be disengaged. This name is the same as that specified for the DTFMR header entry for the file. The operand can be specified either as a symbol or in register notation.

DLINT

```
-------------------------------------------------------
Name        Operation    Operand
-------------------------------------------------------
[name]      DLINT        LFR=n,LINBEG=n
                         [,IMAGE={NO|YES}]
                         [,NOSCAN=(n,...)]
                         [,FLDn=(n,n,NCRIT,xxx)]
                         [,EDITn=(xxxxxx,EDCHAR)]
                         [,FREND={NO|YES}]
-------------------------------------------------------
```

The DLINT macro describes one line type in a format group and the individual fields in the line.

Line Information Entries

LFR=n: This operand specifies the line format record number for the line. The decimal number specified must be in the range of 0 through 63. The line format record describes the format of one type of line; the line format record number is used to identify the line format record. This number is specified in the READ macro when you read a line of data from a document.

LINBEG=n: This operand specifies the beginning of a line. The beginning position is the distance, measured in units of 0.1 inch (2.54 mm), from the left edge of the document to the left boundary of the first field. The limiting range of this position is 4 to 85.

IMAGE={NO|YES}: This operand specifies whether the data record should be in standard mode (IMAGE=NO), or image mode (IMAGE=YES). If this operand is not specified, IMAGE=NO is assumed.

NOSCAN=(n,...): Specifies an area on the document line that is to be ignored by the 3886. 'n' is a decimal number indicating the distance, measured in units of 0.1 inch (2.54 mm), from the left edge of the document to the right end of the NOSCAN field. The field immediately to the left of the NOSCAN field must end with an address delimiter rather than a character delimiter.

Field Information Entries

FLDn=({address-delimiter|character-delimiter},[field-length] [,{NCRIT|font-code|NCRIT,font-code}]):Describes each of the fields in a line. The n in FLDn is a number from 1 through 14 and the parameters are the same for keywords FLD1 through FLD14. The following rules apply when specifying these keywords:

- Fields may be described in any order in the macro.
- Each EDITn parameter must follow its associated FLDn parameter.

- The n suffix need not be 1 for the first field in the line; however, the n suffix must increase for each field from left to right on the document line.

address-delimiter

Is a decimal number that specifies the distance, measured in units of 0.1 inch (2.54 mm), from the left edge of the document to the right end of the field being defined. The last field in a line must end with an address delimiter.

character-delimiter

Specifies the character that indicates the end of a field. The character delimiter is not considered part of the data; it is not included in the data record nor used in determining the length of the field.

Apostrophes enclosing the characters are optional for all characters except 0 through 9, and the special characters used in macro operands. For these characters, the apostrophes are required. For a description of these characters, see OS/VS-DOS/VSE-VM/370 Assembler Language.

If a field ends with a character delimiter, the next field must be read using a font from the same font group. The font groups are:

- NPH1, NPH2, GOTH
- ANA1, ANA2, NUMA, MRKA
- NUMB, MRKB
- ANB1

field-length

Is a decimal number specifying the length of the field in the edited record. The length specified cannot be less than 1 or more than 127. If IMAGE=NO is specified, this parameter is required; if IMAGE=YES is specified, this parameter is invalid. The length specified in this parameter refers to the length of the field after any EDITn options have been performed. The sum of the field lengths for a line cannot be greater than 130.

NCRIT

Indicates that this is not a critical field. If this parameter is omitted, the field is assumed to be critical.

font-code

Specifies a font for this field, different from the font specified in the DFR macro. If this parameter is not specified, the font specified in the DFR macro is used for the field. For information about the valid codes, see the DFR macro description.

EDITn=({code|EDCHAR|code,EDCHAR}): Describes the editing functions to be performed on the data by the 3886.

The parameters are the same for keywords EDIT1 through EDIT14. There must be a FLDn keyword corresponding with each EDITn keyword you specify. If an EDITn keyword is specified, a code, EDCHAR, or both must be specified. When image mode is used, the EDITn keywords are invalid.

When the editing functions are completed and the field is greater than the specified length, the field is truncated from the right and the wrong length field indicator is set on in the header record. If only blanks are truncated, the wrong length field indicator is not set.

code

Specifies the blanks to be removed and the fill characters to be added to the field, if any. The valid codes and their meanings are:

HLBLOF	All high- and low-order blanks are removed, the data is left justified, and the field is padded with blanks on the right (see Note).
ALBLOF	All blanks are removed from the data, the data is left-justified, and the field is padded with blanks on the right. If code is omitted, ALBLOF is assumed.
NOBLOF	No blanks are removed, the data is left-justified, and the field is padded on the right with blanks.
HLBHIF	All high- and low-order blanks are removed, the data is right-justified, and the field is padded to the left with EBCDIC zeros (X'F0') (see Note).
ALBHIF	All blanks are removed, the data is right-justified, and the field is padded with EBCDIC zeros (X'F0') on the left.
ALBNOF	All blanks are removed; the data must be equal in length to the field length specified. No padding is done.

Note: Two consecutive embedded blanks is the maximum number sent.

EDCHAR

Indicates that the characters specified in the EDCHAR keyword of the DFR macro are to be deleted from the field. If this parameter is omitted, the characters are not deleted.

If the EDITn keyword is omitted or if EDITn=EDCHAR is specified and the code is omitted, ALBLOF is assumed.

FREND={NO|YES}: Indicates whether this is the last DLINT macro for the format record. NO indicates that more DLINT macros follow; YES indicates that this is the last one. If this operand is omitted, NO is assumed.

DRMOD

```
-----------------------------------------------------
Name         Operation     Operand
-----------------------------------------------------
[name]       DRMOD         [,DEVICE=3886]
                           [,RDONLY=YES]
                           [,SEPASMB=YES]
                           [,SETDEV=YES]
-----------------------------------------------------
```

The DRMOD macro defines a logic module for a 3886 file. If you do not provide a name for the module, IOCS generates a standard module name.

DEVICE=3886: Specifies that the 3886 is the input device. This operand may be omitted.

RDONLY=YES: This operand generates a read only module. RDONLY=YES must also be specified in the DTF. For additional programming requirements concerning this operand, see the DTFDR RDONLY operand.

SEPASMB=YES: Must be specified if the I/O module is to be assembled separately. This entry causes a CATALR card to be punched preceding the module.

SETDEV=YES: Is specified if the SETDEV macro may be used when processing a file with this I/O module. If SETDEV=YES is specified in the DRMOD macro but not in the DTFDR macro, the SETDEV macro cannot be used when processing that file.

Standard DRMOD Names

Each name consists of eight characters. They are: IJMZxxD0. The fifth and sixth characters are variables as follows:

- If SETDEV=YES is specified, the fifth character is S; otherwise it is Z.
- If RDONLY=YES is specified, the sixth character is R; otherwise it is Z.

Note: Subsetting/supersetting is allowed with the SETDEV keyword, but not with the RDONLY keyword.

DSPLY

Name	Operation	Operand
[name]	DSPLY	{filename|(1)},(r2),(r3)

The DSPLY macro displays the document field on the 1287 display scope. A complete field may be keyboard-entered if a 1287 read error makes this type of correction necessary. An unreadable character may be replaced by the reject character either by the operator (if processing in the on-line correction mode) or by the device (if processing in the off-line correction mode). You may then use the DSPLY macro to display the field in error.

filename|(1): Is the symbolic name specified in the DTFOR header entry for the 1287 file.

(r2): Specifies a general-purpose register (any from 2 to 12) into which the problem program places the address of the load format CCW giving the document coordinates for the field to be displayed. When the DSPLY macro is used in the COREXIT routine, the address of the load format CCW can be obtained by subtracting 8 from the 3-byte address that is right-justified in the fullword location beginning at filename+32. (The high-order fourth byte of this fullword should be ignored.) If the DSPLY macro is not used in the COREXIT routine, you must determine the load format CCW address.

(r3): Specifies a general-purpose register (2 through 12) into which you place the address of the load format CCW giving the coordinates of the reference mark associated with the displayed field.

DTFCD

```
-----------------------------------------------------------------
 Name         Operation    Operand
-----------------------------------------------------------------
 [name]       DTFCD        DEVADDR=SYSxxx
                           ,IOAREA1=name
                           [,ASOCFLE=filename]
                           [,BLKSIZE=n]
                           [,CONTROL=YES]
                           [,CRDERR=RETRY]
                           [,CTLCHR={ASA|YES}]
                           [,DEVICE=nnnn]
                           [,EOFADDR=name]
                           [,ERROPT={IGNORE|SKIP|name}]
                           [,FUNC=xxx]
                           [,IOAREA2=name]
                           [,IOREG=(r)]
                           [,MODE=xx]
                           [,MODNAME=name]
                           [,OUBLKSZ=n]
                           [,RDONLY=YES]
                           [,RECFORM={FIXUNB|VARUNB|UNDEF}]
                           [,RECSIZE=(r)]name]
                           [,SEPASMB=YES]
                           [,SSELECT=n]
                           [,TYPEFLE={INPUT|OUTPUT|CMBND}]
                           [,WORKA=YES]
-----------------------------------------------------------------
```

This macro defines a file for a card reader or a 3881 Optical Mark Reader.

If not stated otherwise, the operands of the DTFCD macro can be specified for all three types of files (INPUT, OUTPUT, CMBND).

ASOCFLE=filename: This operand is used together with the FUNC operand to define associated files for the 2560, 3525, or 5424/5425. (For a description of associated files see the VSE/Advanced Functions Application Programming: Macro User's Guide.) ASOCFLE specifies the filename of associated read, punch, or print files (see Figure 7 on page 57), and enables macro sequence checking by the logic module of each associated file. One filename is required per DTF for associated files.

This operand applies to input and output files.

BLKSIZE=n: Enter the length of the I/O area (IOAREA1). If the record format is variable or undefined, enter the length of the largest record. If the operand FUNC=I is specified for the 2560 or 3525, the length specified for BLKSIZE must be 80 data bytes if CTLCHR=YES or if ASA is not specified; if CTLCHR=YES or if ASA is specified, the length must be 81 bytes.

Code in FUNC= Operand	Filename Specification in ASOCFLE=Operand of		
	Read DTFCD	Punch DTFCD	Print DTFPR
FUNC=PW		Filename of print DTFPR	Filename of punch DTFCD
FUNC=RP	Filename of punch DTFCD	Filename of read DTFCD	
FUNC=RPW	Filename of punch DTFCD	Filename of print DTFPR	Filename of read DTFCD
FUNC=RW	Filename of print DTFPR		Filename of read DTFCD

Examples:

1. If FUNC=PW is specified
   a. specify the filename of the print DTFPR in the ASOCFLE operand of the punch DTFCD and
   b. specify the filename of the punch DTFCD in the ASOCFLE operand of the print DTFPR.

2. If FUNC=RPW is specified
   a. specify the filename of the punch DTFCD in the ASOCFLE oper- and of the read DTFCD , and
   b. specify the filename of the print DTFPR in the ASOCFLE operand of the punch DTFCD, and
   c. specify the filename of the read DTFCD in the ASOCFLE operand of the print DTFPR.

Figure 7. ASOCFLE Operand Usage With Print Associated Files

CONTROL=YES: This operand is specified if a CNTRL macro is to be issued for a file. If this operand is specified, CTLCHR must be omitted.

The CNTRL macro cannot be used for an input file with two I/O areas (that is, when the IOAREA2 operand is specified), or for an input file used in association with a punch file (when the operand FUNC=RP or RPW is specified) on the 2560, 3525, or 5424/5425; in this case, however, this operand can be specified in the DTFCD for the associated punch file.

CRDERR=RETRY: This operand applies to card output on the 2520 or 2540. It specifies the operation to be performed if an error is detected. From this specification, IOCS generates a retry routine and a save area for the card punch record.

If a punching error occurs, it is usually ignored and operation continues. The error card is stacked in stacker P1 (punch), while correct cards are stacked in the stacker you select. If the CRDERR=RETRY operand is included and an error condition occurs, IOCS also notifies the operator and then enters the wait state. The operator can either cancel the job, ignore the error, or instruct IOCS to repunch the card.

CTLCHR={ASA|YES}: This operand is required if first-character control is to be used on an output file. ASA denotes the American National Standards character set. YES denotes the System/370 character set. See Appendix A for the complete list of codes. If this operand is specified, CONTROL must be omitted.

DEVADDR={SYSIPT|SYSPCH|SYSRDR|SYSnnn}: This operand specifies the logical unit name to be associated with a file. The logical unit represents an actual I/O device address and is used in the ASSGN job control statement to assign an actual I/O device address to the file.

SYSIPT, SYSPCH, or SYSRDR must not be specified:

- for the 2596
- for the 3881
- for 1442, 2520, or 2540 combined files (TYPEFLE=CMBND)
- for 2560, 3525, or 5424/5425 associated files (FUNC=RP, RW, RPW, or PW)
- if the operand FUNC=I is specified
- if the MODE operand is specified with the C, O, or R parameters.

DEVICE={2540|1442|2501|2520|2560P|2560S|2596|3504|3505|3525|5425P|5425S|3881}: This operand specifies the I/O device associated with a file. The 'P' and 'S' included with the '2560' and '5425' device codes specify primary or secondary input hoppers. Specify 5425P/S for 5424P/S. If the operand is omitted, 2540 is assumed.

EOFADDR=name: This entry must be included for input and combined files and specifies the symbolic name of your end-of-file routine. IOCS automatically branches to this routine on an end-of-file condition. In your routine you can perform any operations required for the end of the file (you generally issue a CLOSE instruction for the file).

IOCS detects end-of-file conditions in the card reader by recognizing the characters /* punched in card columns 1 and 2 (column 3 must be blank). If the system logical units SYSIPT and SYSRDR are assigned to a 5424/5425, IOCS requires that the /* card, indicating end-of-file, be followed by a blank card. An error condition results if cards are allowed to run out without a /* trailer card (and without a /& card to indicate end-of-job).

ERROPT={IGNORE|SKIP|name}: This operand specifies the error exit option used for an input or output file on a 2560, 3504, 3505, 3525, or 5424/5425. IGNORE, SKIP, or the symbolic name of an error routine can be specified for input files. For output files, only

IGNORE can be specified. This operand must be omitted when using 2560 or 5424/5425 associated output files. The functions of these operands are described below.

IGNORE

The error is to be ignored. When control returns to your program, register 1 contains the address of the error record and, for output files, byte 3, bit 3 of the CCB is set on (see Figure 3 on page 16). You can check this bit and take the appropriate action to recover from the error. Only one I/O area and no work area is permitted for output files. When IGNORE is specified for an input file associated with a punch file (FUNC=RP or RPW) and an error occurs, a PUT for the card in error must nevertheless be given for the punch file.

SKIP

The record in error is not to be made available for processing. The next card is read and processing continues.

name

IOCS branches to your routine when an error occurs, where you may perform whatever actions you desire. Register 1 contains the address of the record in error, and register 14 contains the return address. GET macros must not be issued in the error routine for cards in the same device (or in the same card path for the 2560 or 5424/5425). If the file is an associated file, PUT macros must not be issued in the error routine for cards in the same device (for the 2560 or 5424/5425 this applies to cards in either card path). If any other IOCS macros are issued in the routine, register 14 must be saved. If the operand RDONLY=YES is specified, register 13 must also be saved. At the end of your routine, return to IOCS by branching to the address in register 14. If the input file is associated with an output file (FUNC=RP, RPW, or RW), no punching or printing must be done for the card in error. IOCS continues processing by reading the next card.

> **Note:** When ERROPT is specified for an input file and an error occurs, there is a danger that the /* end-of-file card may be lost. This is because IOCS, after taking the action for the card in error specified by the ERROPT operand, returns to normal processing by reading the next card which is assumed to be a data card. If this card is in fact an end-of-file card, the end-of-file condition cannot be recognized.

FUNC={R|P|I|RP|RW|RPW|PW}: This operand specifies the type of input or output file to be processed by the 2560, 3525, or 5424/5425.

R indicates read.

P indicates punch.

W indicates write (print).

When FUNC=I is specified, the file will be both punched and interpreted; no associated file is necessary to achieve this. The information printed will be the same as the information punched, in contrast to FUNC=PW, where any relation between the information printed and the information punched is determined by your program. When FUNC=I is specified the file can have only one I/O area.

RP, RW, RPW, and PW are used, together with the ASOCFLE operand, to specify associated files; when one of these parameters is specified for one file, it must also be specified for the associated file(s). Each of the associated files can have only one I/O area.

IOAREA1=name: This operand specifies the name of the input or output area used for this file.

If issued for a combined file, this operand specifies the input area. If IOAREA2 is not specified, the area specified in this operand is used for both input and output.

IOAREA2=name: This operand specifies the name of a second I/O area. If the file is a combined file and the operand is specified, the designated area is an output area.

If this operand is specified for the 3881, the IOREG operand must also be specified.

This operand must not be specified if, for the FUNC operand, any of the parameters I, RP, RPW, RW, or PW is specified or if, for an output file, ERROPT=IGNORE is specified.

IOREG=(r): If two input or output areas are used instead of a work area, this operand specifies the register (any of 2 through 12) into which IOCS puts the address of the record. For output files, IOCS puts into this register the address where the user can build a record. This operand cannot be used for combined files.

This operand must be specified for the 3881 if the IOAREA2 operand is specified.

MODE={E|C|O|R|EO|ER|CO|CR}: This operand specifies the mode used to process an input or output file for a 2560, 3504, 3505, or 3525.

E = normal EBCDIC mode, which is also the default.

C = column binary mode.

O = optical mark read (OMR) mode.

R = read column eliminate mode.

E is also assumed if only O or R is specified.

Valid entries are:

- For the 2560: E and C.
- For the 3504 and 3505: E, C, O, R, EO, ER, CO, and CR.
- For the 3525: E, C, R, ER, and CR.
- For SYSIPT, SYSPCH, or SYSRDR: E. O and R (with or without E or C) cannot be specified for output files.

If O or R is specified (with or without E or C), a format descriptor card defining the card columns to be read, or eliminated, must be provided. See OMR considerations in the VSE/Advanced Functions Application Programming: Macro User's Guide for instructions on how to write this card as well as on how to code and process OMR data.

MODNAME=name: This operand is used to specify the name of the logic module that is used with the DTF table to process the file. If the logic module is assembled with the program, MODNAME must specify the same name as the CDMOD macro.

If this operand is omitted, standard names are generated for calling the logic module. If two DTF macros call for different functions that can be handled by a single module, only one module is called.

OUBLKSZ=n: This operand is used in conjunction with IOAREA2, but only for a combined file. Enter the maximum number of characters to be transferred at one time. If this entry is not included and IOAREA2 is specified, the same length as defined by BLKSIZE is assumed.

RDONLY=YES: This operand is specified if the DTF is used with a read-only module. Each time a read-only module is entered, register 13 must contain the address of a 72-byte doubleword-aligned save area. In the case of double buffering support for the 2501 Card Reader, the save area must be 76 bytes to include bytes 0 to 3 of the second CCB generated.

Each task requires its own uniquely defined save area, and each time an imperative macro (except OPEN or OPENR) is issued, register 13 must contain the address of the save area associated with that task. The fact that the save areas are unique for each task makes the module reentrant (that is, capable of being used concurrently by several tasks).

If an ERROPT routine issues I/O macros using the same read-only module that caused control to pass to the error routine, your program must provide another save area. One save area is used for the normal I/O operations, and the second for I/O operations in the ERROPT routine. Before returning to the module that entered the ERROPT routine, register 13 must contain the save area address originally specified for the task.

If this operand is omitted, the module generated is not reenterable, and no save area is required.

RECFORM={FIXUNB|VARUNB|UNDEF}: This operand specifies the record format of the file: fixed length, variable length, or undefined. If the record format is fixed unblocked (FIXUNB,) this operand may be omitted. This operand must specify FIXUNB if you also specified one of the following:

```
TYPEFLE=INPUT
TYPEFLE=CMBND
FUNC=I
DEVICE=3881
```

RECSIZE=(r): For undefined records, this operand specifies the register (any one of 2 through 12) that contains the length of the output record. You must load the length of each record into the specified register before you issue the PUT macro for the record.

SEPASMB=YES: Include this operand only if the DTFCD is to be assembled separately. This causes a CATALR card with the filename to be punched ahead of the object deck and defines the filename as an ENTRY point in the assembly. If the operand is omitted, the assembler assumes that the DTF is assembled together with the problem program.

SSELECT=n: This operand specifies the valid stacker-select character for a file. If this entry is not specified, cards are selected into the NR (normal read) or NP (normal punch) stackers. For the 5424/5425, cards from hopper 1 are placed in stacker 1 and cards from hopper 2 are placed in stacker 5 (or 4).

This operand must not be specified for combined files, for files on the 3881, for 2560, 3525, or 5424/5425 read files associated with punch files (FUNC=RP or FUNC=RPW); in this case the SSELECT=n operand may be specified for the associated output file. For further information, see the CNTRL macro.

When this operand is used with a device other than a 1442 or 2596, the program ignores CONTROL=YES with input files.

TYPEFLE={INPUT|OUTPUT|CMBND}: This operand specifies whether a file is input, output, or combined. A combined file can be specified for a 1442 or 2520 or for a 2540 with the punch-feed-read feature. TYPEFLE=CMBND is applicable if both GETs and PUTs are issued for the same card file.

Only TYPEFLE=INPUT can be specified for the 3881. If OUTPUT or CMBND is specified, the DTF defaults to DEVICE=2540 and a non-executable CDMOD logic module is produced. The MNOTE 'IMPROPER DEVICE. 2540 ASSUMED.' is then printed at assembly time. If the operand is omitted, INPUT is assumed.

WORKA=YES: If I/O records are processed in work areas instead of in the I/O areas, specify this operand. You must set up the work area in storage. The address of the work area, or a general-purpose register which contains the address, must be specified in each GET and PUT macro.

If ERROPT=IGNORE is specified for an output file or if DEVICE=3881, WORKA=YES must not be specified.

DTFCN

```
------------------------------------------------------------
Name        Operation    Operand
------------------------------------------------------------
[name]      DTFCN        DEVADDR=SYSxxx
                         ,IOAREA1=name
                         [,BLKSIZE=n]
                         [,INPSIZE=n]
                         [,MODNAME=name]
                         [,RECFORM={FIXUNB|UNDEF}]
                         [,RECSIZE=(r)]
                         [,TYPEFLE={INPUT|OUTPUT|CMBND}]
                         [,WORKA=YES]
------------------------------------------------------------
```

The DTFCN macro defines an input or output file that is processed on a 3210 or 3215 console printer-keyboard, or a display operator console. DTFCN provides GET/PUT logic as well as PUTR logic for a file.

BLKSIZE=n: This operand specifies the length of the I/O area; if the PUTR macro is used (TYPEFLE=CMBND is specified), this operand specifies the length of the output part of the I/O area. For the undefined record format, BLKSIZE must be as large as the largest record to be processed. The length must not exceed 256 characters.

If the console buffering option is specified at system generation time and the device is assigned to SYSLOG, physical IOCS can increase throughput for each actual output record not exceeding 80 characters. This increase in throughput results from starting the output I/O command and returning to the program before output completion. Regardless of whether or not output records are buffered (queued on an I/O completion basis), they are always printed or displayed in a first-in-first-out (FIFO) order.

DEVADDR={SYSLOG|SYSnnn}: This operand specifies the logical unit name associated with the file. DEVADDR=SYSLOG must be specified to obtain partition identification prefixes (BG, F1, F2, F3, ... Fn) for message identification.

DEVADDR=SYSLOG must be specified if your DTFCN macro includes TYPEFLE=CMBND.

INPSIZE=n: This operand specifies the length of the input part of the I/O area for PUTR macro usage.

IOAREA1=name: This operand specifies the name of the I/O area used by the file. For PUTR macro usage, the first part of the I/O area is used for output, and the second part is used for input. The lengths of these parts are specified by the BLKSIZE and INPSIZE operands respectively. The I/O area is not cleared before or after a message is printed, or when a message is canceled and reentered on the console.

MODNAME=name: This operand specifies the name of the logic module generated by this DTFCN macro. If this entry is omitted, standard module names are generated for the logic module.

A module name must be given when two phases (each containing a DTFCN macro) are link-edited into the same program. Under such conditions, omission of this operand results in unresolved address constants.

RECFORM={FIXUNB|UNDEF}: This operand specifies the record format of the file: fixed length or undefined. FIXUNB must be specified if TYPEFLE=CMBND is specified. FIXUNB is assumed if the RECFORM operand is omitted.

RECSIZE=(r): For undefined records, this operand is required for output files and is optional for input files. It specifies a general register (2 to 12) that contains the length of the record. On output, you must load the length of each record into the specified register before you issue a PUT macro. If specified for input files, IOCS provides the length of the record transferred to storage.

TYPEFLE={INPUT|OUTPUT|CMBND}: This operand specifies a file as input, output, or combined. If INPUT is specified, code is generated for both input and output files. If OUTPUT is specified, code is provided for output files only.

CMBND must be specified if you use the PUTR macro. CMBND specifies that coding be generated for both input and output files; in addition, coding is generated to allow usage of the PUTR macro to ensure that messages requiring operator action are not deleted from the console. When CMBND is specified, DEVADDR=SYSLOG must also be specified.

WORKA=YES: This operand indicates that a work area is used with the file. A GET or PUT macro moves the record to or from the work area. A PUTR macro moves the record from **and** to the work area.

DTFDA

```
-----------------------------------------------------------
Name        Operation    Operand
-----------------------------------------------------------
[name]      DTFDA        BLKSIZE=n
                         ,ERRBYTE=name
                         ,IOAREA1=name
                         ,SEEKADR=name
                         ,TYPEFLE={INPUT|OUTPUT}
                         [,AFTER=YES]
                         [,CONTROL=YES]
                         [,DEVADDR=SYSnnn]
                         [,DSKXTNT=n]
                         [,ERREXT=YES]
                         [,FEOVD=YES]
                         [,HOLD=YES]
                         [,IDLOC=name]
                         [,KEYARG=name]
                         [,KEYLEN=n]
                         [,LABADDR=name]
                         [,READID=YES]
                         [,READKEY=YES]
                         [,RECFORM=xxxxxx]
                         [,RECSIZE=(r)]
                         [,RELTYPE={DEC|HEX}]
                         [,SEPASMB=YES]
                         [,SRCHM=YES]
                         [,TRLBL=YES]
                         [,VERIFY=YES]
                         [,WRITEID=YES]
                         [,WRITEKY=YES]
                         [,XTNTXIT=name]
-----------------------------------------------------------
```

The DTFDA macro defines a file for Direct Access Method (DAM) processing. DAM does not support FBA devices.

If not stated otherwise, the operands of the DTFDA macro can be specified for both input and output files.

AFTER=YES: This operand must be included for output files if any records (or an additional record) are written in a file by a formatting WRITE (count, key, and data) following the last record previously written on a track. The remainder of the track is erased. That is, whenever either of the macros

```
    WRITE filename,AFTER

    WRITE filename,RZERO
```

is used in a program, this operand is required.

BLKSIZE=n: This operand indicates the size of the I/O area by specifying the maximum number of characters that are transferred to or from the area at any one time. When undefined, variable length or spanned records are read or written, the area must be large enough to accommodate the largest record.

For details on how to compute n, see VSE/Advanced Functions Application Programming: Macro User's Guide.

IOCS uses this specification to construct the count field of the CCW for reading or writing records.

CONTROL=YES: Include this operand if a CNTRL macro is issued for this file. The CNTRL macro for seeking on a disk allows you to specify a track address on which access movement should begin for the next READ or WRITE macro. While the arm is moving, you may process data and/or request I/O operations on other devices.

DEVADDR=SYSnnn: This operand must specify the symbolic unit (SYSnnn) associated with a file if the symbolic unit is not provided via an EXTENT job control statement. If such a unit is provided, its specification overrides the DEVADDR parameter. This specification, or symbolic unit, represents an actual I/O address and is used in the ASSGN job control statement to assign the actual I/O device address to the file.

> **Note:** EXTENT job control statements provided for DAM must be supplied in ascending order, and the symbolic units for multi-volume files must be assigned in consecutive order.

DSKXTNT=n: This operand indicates the maximum number of extents (up to 256) that are specified for a file. When this operand is used together with FIXUNB, VARUNB, or UNDEF specified in the RECFORM operand, it indicates that a relative ID is used in the SEEKADR and IDLOC locations. If DSKXTNT=n is omitted, a physical ID is assumed in the SEEKADR and IDLOC locations.

If RECFORM=SPNUNB is specified, DSKXTNT is required. If relative addressing is used, the RELTYPE operand must also be specified.

ERRBYTE=name: This operand is required for IOCS to supply indications of exceptional conditions to your program. The name of a 2-byte field (in which IOCS can store the error-condition or status codes) is entered.

ERREXT=YES: This operand enables unrecoverable I/O errors (occurring before a data transfer takes place) to be indicated to your program. This error information is indicated in the bytes named in the ERRBYTE operand and is available after the WAITF macro has been issued.

FEOVD=YES: This operand is specified if code is generated to handle end-of-volume records. It should be specified only when reading a file which was built using DTFSD and the FEOVD macro.

HOLD=YES: This operand provides for the track hold function, which is to be specified at system generation time. If the operand is omitted, the track hold function is not performed. For details, see VSE/Advanced Functions Application Programming: Macro User's Guide.

IDLOC=name: This operand is included if you want IOCS to supply the ID of a record after each READ or WRITE (ID or KEY) is completed. Specify the name of a record reference field in which IOCS is to store the ID. WAITF should be used before referencing this field. Do not specify the same field for IDLOC and SEEKADR.

> **Note**: When the record to be read or written is the last record of the cylinder, an end-of-cylinder indication is posted in ERRBYTE1, bit 2, but the address returned is that of the **first** record of the **next** cylinder. If, in addition, the end-of-volume indication is posted, the address returned in IDLOC is all 1 bits.

IOAREA1=name: This operand must be included to specify the name of the input/output area used for the file. IOCS routines transfer records to or from this area. The specified name must be the same as the name used in the DS instruction that reserves this area of storage.

KEYARG=name: This operand must be included if records are identified by key; that is, if either of the macros

```
    READ filename,KEY

    WRITE filename,KEY
```

is used in a program, this entry and the corresponding KEYLEN operand are required. KEYARG specifies the name of the key field in which you supply the record key to IOCS.

The KEYARG operand is required for formatting WRITE (WRITE filename,AFTER) operations for files containing keys if RECFORM=VARUNB or SPNUNB. It is required also when the macro

```
  READ filename,ID
```

is specified and if KEYLEN is not zero. When record reference is by key, IOCS uses this specification at assembly time to construct the data address field of the CCW for search commands.

KEYLEN=n: This operand must be included if record reference is by key or if keys are read or written. It specifies the number of bytes in each key. All keys must be the same length. If this operand is omitted, IOCS assumes a key length of zero.

If there are keys recorded on DASD and this entry is absent, a WRITE ID or READ ID writes or reads the data portion of the record.

When record reference is by key, IOCS uses this specification to construct the count field of the CCW for this file. IOCS also uses this in conjunction with IOAREA1 to determine where the data field in the I/O area is located.

LABADDR=name: You may require one or more user labels in addition to the standard file label. If so, you must include your own routine to check, or write, the labels. The name of such a routine is specified in this operand. IOCS branches to this routine after it has processed the standard label.

READID=YES: This operand must be included for an input file if, in your program, the macro 'READ filename,ID' is used.

READKEY=YES: This operand must be included for an input file if, in your program, the macro 'READ filename,KEY' is used.

RECFORM={FIXUNB|SPNUNB|UNDEF|VARUNB}: This operand specifies the type of records in the input or output file. The specifications are:

FIXUNB	For fixed-length records. All records are considered unblocked. If you want blocked records, you must provide your own blocking and deblocking.
SPNUNB	For spanned records. This specification is for unblocked variable-length logical records of less than 32,768 bytes per record.
UNDEF	For undefined records. This specification is required only if the records are of undefined format.
VARUNB	For variable-length records. This specification is for unblocked variable-length records.

For a definition of record formats see VSE/Advanced Functions, Data Management Concepts.

RECSIZE=(r): This operand must be included if undefined records are specified (RECFORM=UNDEF). It specifies the number of the general-purpose register (any of 2 through 12) that contains the length of each individual input or output record.

Whenever an undefined record is read, IOCS supplies the length of the data area for that record in the specified register.

When an undefined record is written, you must load the length of the data area of the record (in bytes) into this register, before you issue the WRITE macro for the record. IOCS adds the length of the key when required.

When records are written (AFTER specified in the WRITE macro), IOCS uses the length to construct the count area written on DASD. IOCS adds the length of both the count and the key when required.

RELTYPE={DEC|HEX}: This operand specifies whether the zoned decimal (DEC) or hexadecimal (HEX) form of the relative ID is to be used. When FIXUNB, VARUNB, or UNDEF is specified in the RECFORM operand, RELTYPE should be supplied only if the DSKXTNT operand (relative ID) is specified. If omitted, a hexadecimal relative ID is assumed. However, if DSKXTNT is also omitted, a physical ID is assumed in the SEEKADR and IDLOC addresses.

If RECFORM=SPNUNB is specified, the RELTYPE operand is required when relative addressing is used. If RELTYPE is omitted, a physical ID is assumed in the SEEKADR and IDLOC addresses.

SEEKADR=name: This operand must be included to specify the name of your track-reference field. In this field, you store the track location of the particular record read or written. IOCS refers to this field to determine which volume and which track contains the desired record. Whenever records are to be located by searching for a specified ID, the track-reference field must also contain the number of the record on the track.

SEPASMB=YES: Include this operand only if the DTFDA is to be assembled separately. This causes a CATALR card with the filename to be punched ahead of the object deck and the filename to be defined as an ENTRY point in the assembly. If the operand is omitted, the assembler assumes that the DTF is assembled together with the problem program.

SRCHM=YES: If records are identified by key, this operand may be included to cause IOCS to search multiple tracks for each specified record. The macros

```
    READ filename,KEY

    WRITE filename,KEY
```

cause IOCS to search the track specified in the track-reference field and all following tracks in the cylinder, until the record is found or the end of the cylinder is reached. If the file ends before the end of the cylinder and the record is not found, the search continues into the next file, if any, on the cylinder. EOC, instead of NRF, is indicated. Without SRCHM=YES, each search is confined to the specified track.

TRLBL=YES: This operand, if specified with the LABADDR operand, indicates that user standard trailer labels are to be read or written following the user standard header labels on the user label track. Both operands must be specified for trailer label processing.

TYPEFLE={INPUT|OUTPUT}: This operand must be included to indicate how standard volume and file labels are to be processed. INPUT indicates that standard labels are to be read; OUTPUT indicates that standard labels are to be written.

VERIFY=YES: This operand is included if you want to check the parity of disk records after they are written. If this operand is omitted, any records written on a disk are not verified.

WRITEID=YES: This operand must be included if the DASD storage location for writing any output record or updating an input file is specified by a record ID (identifier); that is, whenever the macro

```
  WRITE filename,ID
```

is used in the program, this operand is required.

WRITEKY=YES: This operand must be included if the DASD location for writing any output record or updating an input file is specified by record key, that is, whenever

```
  WRITE filename,KEY
```

is used.

XTNTXIT=name: This operand is included if you want to process label extent information. It specifies the name of your extent exit routine. During an OPEN, IOCS branches to your routine after each specified extent is checked. Upon entering your routine, IOCS stores, in register 1, the address of a 14-byte field that contains the label extent information (in binary form) retrieved from the format 1 and format 3 labels. If user labels are present, the user label track is returned as a separate extent and the lower limit of the first normal extent is increased by one track. The format of this field is shown in Figure 8 . Return to IOCS by use of the LBRET macro. Registers 2 through 13 are available in the XTNTXIT routine. Within the routine you cannot issue a macro that calls a transient routine (such as OPEN, CLOSE, DUMP, PDUMP, CANCEL, CHKPT, etc.).

Bytes	Contents
0	Extent type code (as specified in the EXTENT statement).
1	Number of extent (as determined by the EXTENT statement sequence).
2–5	Lower limit of the extent (cchh).
6–9	Upper limit of the extent (cchh).
10–11	Symbolic unit number (in hexadecimal format).
12–13	Not used.

Figure 8. Label Extent Information Field

DTFDI

```
-------------------------------------------------------------
Name         Operation    Operand
-------------------------------------------------------------
[name]       DTFDI        DEVADDR=SYSxxx
                          ,IOAREA1=name
                          [,CISIZE=n]
                          [,EOFADDR=name]
                          [,ERROPT={IGNORE|SKIP|name}]
                          [,IOAREA2=name]
                          [,IOREG=(r)]
                          [,MODNAME=name]
                          [,RDONLY=YES]
                          [,RECSIZE=n]
                          [,SEPASMB=YES]
                          [,TRC=YES]
                          [,WLRERR=name]
-------------------------------------------------------------
```

The DTFDI macro provides device independence for system logical units.

CISIZE=n: This operand specifies the FBA control interval size. The value n must be an integral multiple of the FBA physical block size and, if greater than 8K, must be a multiple of 2K. The maximum value is 32768 (32K), except when assigned to SYSLST or SYSPCH, when the maximum is 30720 (30K).

If CISIZE is omitted, CISIZE=0 is assumed. For FBA devices, control interval size may be overridden for an output file at execution time by specifying the CISIZE parameter of the DLBL control statement. For an input file, the CISIZE value in the format-1 label is used. If the CISIZE value is zero, then OPEN calculates a value based on the RECSIZE value specification.

DEVADDR={SYSIPT|SYSLST|SYSPCH|SYSRDR}: This operand must specify the symbolic unit associated with this system file. Only the system names shown above may be specified. The logical device SYSLST must not be assigned to the 2560 or 5424/5425.

EOFADDR=name: This operand specifies the name of your end-of-file routine. It is required only if SYSIPT or SYSRDR is specified.

IOCS branches to this routine when it detects an end-of-file condition. In this routine, you can perform any operations necessary for the end-of-file condition (you generally issue the CLOSE macro).

An end-of-file condition exists when the following occurs for either SYSIPT or SYSRDR:

- for a card reader, a /* in positions 1 and 2 of the record.

- for tape, a /* in positions 1 and 2 of the record or a tapemark.
- for disk, a /* in positions 1 and 2 of the record or an end-of-file record.

If the system logical units SYSIPT and SYSRDR are assigned to a 5424/5425, IOCS requires that the /* card, indicating end-of-file, be followed by a blank card. An error condition results if the records are allowed to run out without a /* card (and without a /& card, if end-of-job). IOCS detects the end-of-file condition on diskette units by recognizing that end-of-data has been reached on the current volume and that there are no more volumes available.

ERROPT={IGNORE|SKIP|name}: This operand does not apply to output files. For output files for most devices, the job is automatically terminated after IOCS has attempted to retry writing the record; for 2560 or 5424/5425 output files, normal error recovery procedures are followed.

This operand applies to wrong-length records if WLRERR is omitted. If both ERROPT and WLRERR are omitted and wrong-length records occur, IOCS ignores the error.

ERROPT specifies the function to be performed for an error block. If an error is detected when reading a magnetic tape, or a disk or a diskette volume, IOCS attempts to recover from the error. If the error is not corrected, the job is terminated unless this operand is included to specify other procedures to be taken. The three specifications are described below.

IGNORE
The error condition is to be ignored. The address of the error record is made available to you for processing (see CCB Macro).

SKIP
The error block is not to be made available for processing. The next record is read and processing continues.

name
IOCS is to branch to your routine when an error occurs, where you may perform whatever functions are desired or simply note the error condition. The address of the error record is supplied in register 1. The contents of the IOREG register may vary and should not be used for error records. Also, you must not issue any GET instructions in your error routine. If you use any other IOCS macros, you must save the contents of register 14. You must also save the contents of register 13. At the end of the error routine, return to IOCS by branching to the address in register 14. The next record is then made available for processing.

IOAREA1=name: This operand must specify the name of the input or output area used with the file. The input and/or output routines transfer records to or from this area.

If the DTFDI macro is used to define a printer file, or a card file to be processed on a 2540, 2560, 3525, or 5424/5425, the first byte of the output area must contain a control character.

IOAREA2=name: Two input or output areas can be allotted for a file to permit overlapped GET or PUT processing. If this operand is included, it specifies the name of the second I/O area.

IOREG=(r): When two I/O areas are used, this operand specifies the general purpose register (any of 2 through 12) that points to the address of the next record. For input files, it points to the logical record available for processing. For output files, it points to the address of the area where you can build a record. If omitted, and two I/O areas are used, register 2 is assumed.

MODNAME=name: This operand may be used to specify the name of the logic module used with the DTF table to process the file. If the logic module (DIMOD) is assembled with the program, the MODNAME parameter in this DTF must specify the same name as the DIMOD macro.

If this entry is omitted, standard names are generated for calling the logic module. If two different DTF macros call for different functions that can be handled by a single module, only one standard-named module is called.

This operand is ignored for all DASD devices. An IBM-supplied module is always used for these devices.

RDONLY=YES: This operand is specified if the DTF is to be used with a read-only module. Each time a read-only module is entered, register 13 must contain the address of a 72-byte doubleword-aligned save area. Each task should have its own uniquely defined save area, and each time an imperative macro (except OPEN, OPENR or LBRET) is issued, register 13 must contain the address of the save area associated with that task. The fact that the save areas are unique for each task makes the module reentrant (that is, capable of being used concurrently by several tasks).

If an ERROPT or WLRERR routine issues I/O macros using the same read-only module that caused control to pass to either error routine, the program must provide another save area. One save area is used for the initial I/O operations, and the second for I/O operations in the ERROPT or WLRERR routine. Before returning to the module that entered the error routine, register 13 must be set to the save area address originally specified for the task.

If the operand is omitted, the module generated is not reenterable and no save area need be established.

This operand is ignored for all DASD devices. For these devices a read-only module is always supplied.

RECSIZE=n: This operand specifies the length of the record. For input files (SYSIPT and SYSRDR), the maximum allowable record size is 81 bytes. You should always specify the maximum of 81 bytes (and an I/O area of 81 bytes) to ensure device independence when reading data. The first byte of the I/O area will always contain the first data byte, regardless of whether the input consisted of 80 data bytes and one control character or of 80 data bytes only. For output files, RECSIZE must include one byte for control characters. The maximum length specification is 121 for SYSLST and 81 for SYSPCH.

For disk files, 121 must be specified for SYSLST, and 81 for SYSPCH. For printers and punches, DIMOD assumes a S/370-type control character if the character is not a valid ASA character. The program checks ASA control characters before S/370-type control characters. Therefore, if it is a valid ASA control character (even though it may also be a S/370-type control character), it is used as an ASA control character. Otherwise, it is used as a S/370-type control character.

Control character codes are listed in Appendix A; note, however:

- 2520 stacker selection codes must be used for the 1442.
- 2540 stacker selection 3 must not be used if device independence is to be maintained.

If this operand is omitted, the following is assumed:

80 bytes for SYSIPT.

80 bytes for SYSRDR.

81 bytes for SYSPCH.

121 bytes for SYSLST.

SEPASMB=YES: Include this operand only if the DTFDI is to be assembled separately. This causes a CATALR card with the filename to be punched ahead of the object deck and defines the filename as an ENTRY point in the assembly. If the operand is omitted, the assembler assumes that the DTF is assembled together with the problem program.

TRC=YES: This operand applies to the IBM 3800 Printing Subsystem. TRC=YES specifies that a table reference character is included as the first byte of each output data line (following the optional print control character). The printer uses the table reference character (0, 1, 2, or 3) to select the character arrangement table corresponding to the order in which the table names have been specified with the CHARS parameter on the SETPRT job control statement (or SETPRT macro instruction).

If the device allocated is not a printer and TRC=YES is specified, the table reference character is treated as data when a PUT is issued. If the device is a non-3800 printer, the table reference character is removed and not printed.

WLRERR=name: This operand applies only to input files on devices other than diskette units. It specifies the name of your routine to which IOCS branches if a wrong-length record is read on a tape or disk device.

If this operand is omitted and a wrong-length error occurs, the ERROPT routine will be invoked if it is available.

DTFDR

```
--------------------------------------------------
Name         Operation    Operand
--------------------------------------------------
[name]       DTFDR        COREXIT=name
                          ,DEVADDR=SYSxxx
                          ,EOFADDR=name
                          ,EXITIND=name
                          ,FRNAME=name
                          ,FRSIZE=n
                          ,HEADER=name
                          ,IOAREA1=name
                          [,BLKSIZE=n]
                          [,DEVICE=3886]
                          [,MODNAME=name]
                          [,RDONLY=YES]
                          [,SEPASMB=YES]
                          [,SETDEV=YES]
--------------------------------------------------
```

You must use the DTFDR macro to define each 3886 file in your program.

BLKSIZE=nnn: Specifies the length of the area named by the IOAREA1 keyword. The length of the area must be equal to the length of the longest record to be passed from the 3886.

If this operand is omitted, the maximum length of 130 is assumed.

> **Note:** LIOCS does not allow you to block records read from the 3886.

COREXIT=name: Provides the symbolic name of your error correction routine. LIOCS branches to this routine whenever an error is indicated in the EXITIND byte.

You can attempt to recover from various errors that occur on the 3886 through the COREXIT routine you provide. Your COREXIT routine receives control whenever one of the following conditions occurs:

- Incomplete scan
- Line mark station timing mark check error
- Nonrecovery error
- Permanent error

> **Note:** If any of these errors occur while the file is being opened, the COREXIT routine does not receive control and the job is canceled.

Figure 9 on page 79 describes normal functions for the COREXIT routine for the various error conditions and provides the exits that must be taken from the COREXIT routine.

Error messages are provided to describe errors to the operator during program execution.

DEVADDR=SYSxxx: Specifies the symbolic unit to be associated with the logical file. The symbolic unit is associated with an actual I/O device through the job control ASSGN statement.

DEVICE=3886: Indicates that 3886 is the I/O device for this file. This operand may be omitted.

EOFADDR=name: Specifies the symbolic address of your end-of-file routine. LIOCS branches to this routine whenever end of file is detected on the 3886.

EXITIND=name: Specifies the symbolic name of the 1-byte area in which the completion code is returned to the COREXIT routine for error handling from an I/O operation.

The completion codes are:

Dec	Hex	Meaning
240	X'F0'	No errors occurred. (This code should not be present when the COREXIT routine receives control.)
241	X'F1'	Line mark station timing mark check error.
242	X'F2'	Nonrecovery error. Do not issue the CNTRL macro to eject the document from the machine. Have the operator remove the document.
243	X'F3'	Incomplete scan.
244	X'F4'	Line mark station timing mark check and equipment check.
249	X'F9'	Permanent error.

Note: If any of these errors occur while the file is being opened, the COREXIT routine does not receive control and the job is canceled.

FRNAME=phasename: Specifies the phase name of the format record to be loaded when the file is opened.

FRSIZE=n: Specifies the number of bytes to be reserved in the DTF expansion for format records. The number must equal at least the size of the largest DFR macro expansion and its associated DLINT macro expansions, plus four. This size is printed in the ninth and tenth bytes of the DFR macro expansion.

If you use the SETDEV macro in your program to change format records, you can reduce the library retrieval time by specifying a size large enough to contain all the frequently used format records.

Error	Normal COREXIT Function	Exit to
X'F2'	Eliminate the data that has been read from this document and prepare to read the next document (see Note 1).	Routine in your program to read the next document.
X'F4' or X'F9'	Do whatever processing is necessary before the job is canceled (see Note 1).	Your end-of-job routine.
X'F1'	Do any processing that may be required. The document may have been read incorrectly; you may want to delete all data records read in (see Note 2).	Branch to the address in reg. 14 to return to the instruction following the macro causing the error.
X'F3'	Rescan the line using another format record or using image processing and editing the record in your program (see Note 2).	Branch to the address in reg. 14 to return to the instruction following the macro causing the error.

Notes:

1. If, in your COREXIT routine, you issue an I/O macro to the 3886 and an error occurs during that operation, control is returned to the beginning of the COREXIT routine. You must take precautions in the COREXIT routine to prevent looping in this situation. If no errors occur control returns to the instruction following the I/O macro.

2. If, in your COREXIT routine, you issue an I/O macro to the 3886, control always returns to the instruction following the macro. You should then check the completion code to determine the outcome of the operation.

Figure 9. COREXIT Routine Functions

The area should then be equal to the sum of the format record sizes, plus four bytes for each format record. When the SETDEV macro is issued, the format record is loaded into this area from the core image library if it is not already present in the area.

HEADER=name: Specifies the symbolic name of the 20-byte area to receive the header record from the 3886.

IOAREA1=name: Specifies the symbolic name of the input area to be used for the file. The area must be as large as the size specified in the BLKSIZE parameter. If BLKSIZE is not specified, the input area must be 130 bytes.

MODNAME=name: This operand may be used to specify the name of the logic module used with the DTF table to process the file. If the logic module (DRMOD) is assembled with the program, the MODNAME parameter in this DTF must specify the same name as the DRMOD macro.

If this entry is omitted, standard names are generated for calling the logic module. If two different DTF macros call for different functions that can be handled by a single module, only one standard-named module is called.

RDONLY=YES: This operand is specified if the DTF is used with a read-only module. Each time a read-only module is entered, register 13 must contain the address of a 72-byte doubleword-aligned save area. Each DTF should have its own uniquely defined save area.

Each time an imperative macro (except OPEN or OPENR) is issued using a particular DTF, register 13 must contain the address of the save area associated with that DTF.

If a COREXIT routine issues I/O macros using the same read-only module that caused control to pass to either error routine, your program must provide another save area. One save area is used for the normal I/O operations, and the second for I/O operations in the COREXIT routine. Before returning to the module that entered the COREXIT routine, register 13 must contain the save area address originally specified for that DTF.

If this operand is omitted, the module generated is not reenterable, and no save area is required.

SEPASMB=YES: Specifies that the DTF is to be assembled separately. If this operand is specified, a CATALR card with the filename is punched before the deck and the filename is defined as an ENTRY point for the assembly.

SETDEV=YES: Specifies that the SETDEV macro is issued in your program to load a different format record into the 3886.

DTFDU

```
------------------------------------------------------------
Name        Operation    Operand
------------------------------------------------------------
[name]      DTFDU        EOFADDR=name
                         ,IOAREA1=name
                         ,RECSIZE=n
                         [,CMDCHN=n]
                         [,DEVADDR=SYSxxx]
                         [,DEVICE=3540]
                         [,ERREXT=YES]
                         [,ERROPT={IGNORE|SKIP|name}]
                         [,FEED={YES|NO}]
                         [,FILESEC=YES]
                         [,IOAREA2=name]
                         [,IOREG=(r)]
                         [,MODNAME=name]
                         [,RDONLY=YES]
                         [,SEPASMB=YES]
                         [,TYPEFLE={INPUT|OUTPUT}]
                         [,VERIFY=YES]
                         [,VOLSEQ=YES]
                         [,WORKA=YES]
                         [,WRTPROT=YES]
------------------------------------------------------------
```

The DTFDU macro defines sequential (consecutive) processing for a file contained on a diskette.

CMDCHN=n: This operand is specified to indicate the number of Read/Write CCWs to be command chained. Valid entries are 1, 2, 13, or 26; 1 is assumed if this operand is omitted. For each CCW specified by this operand, one record is processed (for example, if you code CMDCHN=13, 13 records are command chained and are processed - read or written - as a group). For entries of 2, 13, or 26, either the IOREG operand or the WORKA operand must be specified.

DEVADDR=SYSxxx: This operand specifies the symbolic unit (SYSxxx) associated with the file if an EXTENT job control statement is not provided. An EXTENT statement is not required for single-volume input files. If an EXTENT statement is provided, its specification overrides any DEVADDR specification. SYSxxx represents an actual I/O device address, and is used in the ASSGN job control statement to assign the actual I/O device address to this file.

DEVICE=3540: This operand specifies that the file to be processed is on the 3540. This operand may be omitted.

EOFADDR=name: This operand specifies the symbolic name of your end-of-file routine. IOCS automatically branches to this routine on an end-of-file condition. You can perform any operations required for the end-of-file in this routine (you will generally issue the CLOSE macro).

ERREXT=YES: This operand enables your ERROPT routine to return to DUMODFx with the ERET macro. It also enables permanent errors to be indicated to your program. For ERREXT facilities, the ERROPT operand must be specified. However, to take full advantage of this option, use the ERROPT=name operand.

ERROPT={IGNORE|SKIP|name}: Specify this operand if you do not want a job to be terminated when a permanent error cannot be corrected in the diskette error routine. If attempts to reread a chain of records are unsuccessful, the job is terminated unless the ERROPT entry is included. Either IGNORE, SKIP, or the name of an error routine can be specified. The functions of these parameters are described below.

IGNORE
The error condition is ignored. The records are made available for processing. On output, the error condition is ignored and the records are considered written correctly.

SKIP
No records in the error chain are made available for processing. The next chain of records is read from the diskette, and processing continues with the first record of that chain. On output, the SKIP option is the same as the IGNORE option.

name
IOCS branches to your error routine named by this parameter regardless of whether or not ERREXT=YES is specified. In this routine you can process or make note of the error condition as desired.

If ERREXT is not specified, register 1 contains the address of the first record in the error chain. When processing in the ERROPT routine, reference records in the error chain by referring to the address supplied in register 1. The contents of the IOREG register or work area are variable and should not be used to process error records. Also, GET macros must not be issued for records in the error chain. If any other IOCS macros (excluding ERET if ERREXT=YES) are used in this routine, the contents of register 13 (with RDONLY) and 14 must be saved and restored after their use. At the end of the routine, return control to IOCS by branching to the address in register 14. For a read error, IOCS skips that error chain of records, and makes the first record of the next chain available for processing in the main program.

If ERREXT is specified, register 1 contains the address of a two part parameter list containing the 4-byte DTFDU address and the 4-byte address of the first record in the error chain. Register 14 contains the return address. Processing is similar to that described above except for addressing the records in error.

At the end of its processing, the routine returns to LIOCS by issuing the ERET macro.

For an input file, the program:

- skips the error chain and reads the next chain with an ERET SKIP,
- ignores the error with an ERET IGNORE,
- it makes another attempt to read the error chain with an ERET RETRY.

For an output file the only acceptable parameters are IGNORE or name, and the program

- ignores the error condition with ERET IGNORE or ERET SKIP,
- attempts to write the error chain with an ERET RETRY. Bad spot control record (1, 2, 13, or 26 records depending on the CMDCHN specification) are written at the current diskette address, and the write chain is retried in the next 1, 2, 13, or 26 (depending on the CMDCHN specification) sectors on the disk.

The DTFDU error options are shown in Figure 10 .

To terminate the job,	specify nothing.
To skip the error record,	specify ERROPT=SKIP.
To ignore the error record,	specify ERROPT=IGNORE.
To process the error record,	specify ERROPT=name.
After processing the record, to leave the error-processing routine and	
To skip the (input) record,	execute ERET SKIP.
To ignore the record,	execute ERET IGNORE.
To retry reading or writing the record,	execute ERET RETRY.

Figure 10. DTFDU Error Options

FEED={YES|NO}: If YES is specified and IOCS detects an end-of-file condition, the diskette being processed is fed to the stacker and a new diskette is fed to the disk drive (providing another diskette is still in the hopper). If NO is specified, the diskette is left mounted for the next job. If the operand is omitted, YES is assumed.

FILESEC=YES: This operand applies to output only. On output it causes OPEN to set the security flag in the file label. For subsequent input, the security flag causes an operator message to be written. The operator must then reply in order to make the file available to be read.

> **Note:** When this operand is used with WRTPROT=YES, the reuse of the diskette is prevented.

IOAREA1=name: This operand specifies the symbolic name of the I/O area used by the file. IOCS either reads or writes records using this area. Note that you should provide an I/O area equal in size to the result obtained from multiplying the RECSIZE entry by the CMDCHN entry.

IOAREA2=name: If two I/O areas are used by GET or PUT, this operand is specified. You should provide an I/O area equal in size to the result obtained from multiplying the RECSIZE entry by the CMDCHN entry.

IOREG=(r): This operand specifies the general purpose register (any one of 2 to 12) in which IOCS puts the address of the logical record that is available for processing. At OPEN time, for output files, IOCS puts the address of the area where the user can build a record in this register. The same register can be used for two or more files in the same program, if desired. If this is done, the problem program must store the address supplied by IOCS for each record. If this operand is specified, omit the WORKA operand.

This operand must be specified if the CMDCHN factor is 2 or higher and records are processed in one I/O area, or if two I/O areas are used and records are processed in both I/O areas.

MODNAME=name: This operand specifies the name of the logic module which is to process the file. If the logic module is assembled with the program, MODNAME must specify the same name as the DUMODFx macro. If this operand is omitted, standard names are generated for calling the logic module. If two DTF macros call for different functions that can be handled by a single module, only one module is called.

RDONLY=YES: This operand is specified if the DTF is used with a read-only module. Each time a read-only module is entered, register 13 must contain the address of a 72-byte double-word aligned save area. Each task should have its own uniquely defined save area. When an imperative macro (except OPEN, OPENR) is issued, register 13 must contain the address of the save area associated with the task. The fact that the save areas are unique for each task makes the module reentrant (that is, capable of being used concurrently by several tasks).

If an ERROPT routine issues I/O macros using the same read-only module that caused control to pass to the error routine, your problem program must provide another save area. One save area is used for the normal I/O operations, and the second for input/output operations in the ERROPT routine. Before returning to the module that entered the ERROPT routine, register 13 must be set to the save area address originally specified for that DTF.

If this operand is omitted, the generated module is not reentrant and no save area need be established.

RECSIZE=n: This operand specifies (in bytes) the length of each record in the input/output area (1 to 128 bytes).

SEPASMB=YES: Include this operand only if the DTFDU is to be assembled separately. This causes a CATALR card with the filename to be punched ahead of the object deck and the filename to be defined as an entry point in the assembly. If the operand is omitted, the assembler assumes that the DTF is assembled together with the problem program.

TYPEFLE={INPUT|OUTPUT}: This operand indicates whether the file is an input or output file.

VERIFY=YES: This operand specifies that the input on a 3741/3742 must be verified before processing may continue. If VERIFY=YES is not specified, it is assumed that the input need not be verified. If VERIFY=YES is specified and the input is not verified, the job is canceled and message 4n57I is issued. If the operand is specified for an output file, it will be ignored.

VOLSEQ=YES: This operand is only valid on input. If specified, it causes OPEN to ensure that the volume sequence numbers of a multi-volume file are in ascending and sequential order. However, if the volume sequence number of the first volume processed is blank, no volume sequence checking is done.

WORKA=YES: If I/O records are processed or built in work areas instead of in the I/O areas, specify this operand. You must set up the work area in storage. The address of the work area, or a general register containing the address, must be specified in each GET or PUT macro. For a GET or PUT macro, IOCS moves the record to or from the specified work area.

When this operand is specified, the IOREG operand must be omitted.

WRTPROT=YES: This operand indicates that an output file will be created with Write-Protect (meaning that the file cannot be overwritten). For 3540 support, this has no effect on subsequent input processing of the file.

> **Note:** When this operand is used with FILESEC=YES, the reuse of the diskette is prevented.

```
-------------------------------------------------------------
Name        Operation    Operand
-------------------------------------------------------------
[name]      DTFIS        DSKXTNT=n
                         ,IOROUT=xxxxxx
                         ,KEYLEN=n
                         ,NRECDS=n
                         ,RECFORM={FIXUNB|FIXBLK}
                         ,RECSIZE=n
                         [,CYLOFL=n]
                         [,DEVICE=nnnn]
                         [,ERREXT=YES]
                         [,HINDEX=nnnn]
                         [,HOLD=YES]
                         [,INDAREA=name]
                         [,INDSKIP=YES]
                         [,INDSIZE=n]
                         [,IOAREAL=name]
                         [,IOAREAR=name]
                         [,IOAREAS=name]
                         [,IOAREA2=name]
                         [,IOREG=(r)]
                         [,IOSIZE=n]
                         [,KEYARG=name]
                         [,KEYLOC=n]
                         [,MODNAME=name]
                         [,MSTIND=YES]
                         [,RDONLY=YES]
                         [,SEPASMB=YES]
                         [,TYPEFLE={RANDOM|SEQNTL|RANSEQ}]
                         [,VERIFY=YES]
                         [,WORKL=name]
                         [,WORKR=name]
                         [,WORKS=YES]
-------------------------------------------------------------
```

The DTFIS macro defines a DASD file for the Indexed Sequential Access Method.

CYLOFL=n: This operand must be included if cylinder overflow areas are reserved for a file. Do not include this entry if no overflow areas are reserved.

When a file is loaded or when records are added, this operand is required to reserve the areas for cylinder overflow. It specifies the number of tracks to be reserved on each cylinder. The maximum number of tracks that can be reserved on each cylinder is:

for 2311	8
for 2314, or 2319	18
for 3330 or 3333	17
for 3340	10

DEVICE={2311|2314|3330|3340}: This operand specifies the unit that contains the prime data area and overflow areas for the logical file. For ISAM the prime data area and the overflow areas must be on the same device type, and, for a 3340, the data modules must be of the same size (35 or 70MB). If the operand is omitted, 2311 is assumed.

DSKXTNT=n: This operand must be included to specify the maximum number of extents for this file. The number must include all the data area extents if more than one DASD area is used for the data records, and all the index area and independent overflow area extents that are specified by EXTENT job control statements. Thus the minimum number specified by this entry is 2: one extent for one prime data area, and one for a cylinder index. Each area assigned to an ISAM file is considered an extent.

> **Note:** Master and cylinder indexes are treated as one area. When there is one master index extent, one cylinder index extent, and one prime data area extent, DSKXTNT=2 could be specified.

ERREXT=YES: This operand is required for IOCS to supply your program with detailed information about unrecoverable I/O errors occurring before a data transfer takes place, and for your program to be able to use the ERET imperative macro to return to IOCS specifying an action to be taken for an error condition.

Some error information is available for testing by your program after each imperative macro is executed, even if ERREXT=YES is not specified, by referencing field filenameC. Filename is the same name as that specified in the DTF header entry for the file. One or more of the bits in the filenameC byte may be set to 1 by IOCS. The meaning of the bits varies depending on which parameter was specified in the IOROUT operand; Figure 11 on page 89 shows the meaning if IOROUT=ADD, RETRVE, or ADDRTR was specified; Figure 12 on page 90 shows the meaning if IOROUT=LOAD was specified.

If ERREXT=YES is not specified, IOCS returns the address of the DTF table in register 1, as well as any data-transfer error information in filenameC, after each imperative macro is executed; non-data-transfer error information is not given. After testing filenameC, return to IOCS by issuing any imperative macro except ERET; no special action is taken by IOCS to correct or check an error.

If ERREXT=YES is specified, IOCS returns the address of an ERREXT parameter list in register 1 after each imperative macro is executed, and information about both data-transfer and non-data-transfer errors in filenameC. The format of the ERREXT parameter list is shown in Figure 13 on page 91 . After testing filenameC and finding an error, return to IOCS by using the ERET imperative macro; IOCS takes the action indicated by the ERET operand. If HOLD=YES (and ERREXT=YES), ERET must be used to return to IOCS to free any held track.

In your program, you should check byte 16, bit 7 of the DTF for a blocksize compatibility error when adding to, or extending a file. If the blocksize of your program is not equal to the blocksize of the previously built file, this bit will be set to 1.

HINDEX={2311|2314|3330|3340}: This entry specifies the unit containing the highest index.

Placing the highest index on a separate unit is recommended only if that unit is physically separate from the unit(s) holding the track indexes and the data of the file, and if it has its own access mechanism. If this operand is omitted, 2311 is assumed.

HOLD=YES: This operand provides for the track hold option for both data and index records. If the HOLD operand is omitted, the track hold function is not performed. Because track hold cannot be performed on a LOAD file, HOLD=YES cannot be specified when IOROUT=LOAD.

If HOLD=YES and ERREXT=YES, your program must issue the ERET macro to return to the ISAM module to free any held tracks.

INDAREA=name: This operand specifies the name of the area assigned to the cylinder index. If specified, all or part of the cylinder index resides in virtual storage thereby increasing throughput. If this operand is included, INDSIZE must be included.

If the area assigned to INDAREA is large enough for all the index entries to be read into virtual storage at one time and the index skip feature (INDSKIP) is not specified, no presorting of records need be done. If the area assigned to INDAREA is not large enough, the records processed should be presorted to fully utilize the resident cylinder index.

INDSKIP=YES: When cylinder index entries reside in virtual storage, this operand specifies the index skip feature. This feature allows ISAM to skip any index entries preceding those needed to process a given key. If the index skip operand is omitted, the cylinder indexes are processed sequentially.

This operand may be specified only with the INDAREA and INDSIZE operands and increases throughput only when:

- The records are presorted.
- The allocated virtual storage is insufficient for storing all of the cylinder index.
- One or more large segments of the file are not referenced.

Bit	Cause	Explanation
0	DASD error	An uncorrectable DASD error has occurred (except wrong length record.)
1	Wrong length record	A wrong length record has been detected during an I/O operation.
2	End of file	The EOF condition has been encountered during execution of the sequential retrieval function.
3	No record found	The record to be retrieved has not been found in the file. This applies to RANDOM (RANSEQ) and to SETL in SEQNTL (RANSEQ) when KEY is specified, or after GKEY. This may also be a hardware error.
4	Illegal ID specified	The ID specified to the SETL in SEQNTL (RANSEQ) is outside the prime file limits.
5	Duplicate record	The record to be added to the file has a duplicate record key of another record in the file.
6	Overflow area full	An overflow area in a cylinder is full, and no independent overflow area has been specified; or an independent overflow area is full, and the addition cannot be made. You should assign an independent overflow area or extend the limit.
7	Overflow	The record being processed in one of the retrieval functions (RANDOM/SEQNTL) is an overflow record.

Figure 11. FilenameC-Status or Condition Code Byte if IOROUT=ADD, RETRVE, or ADDRTR

INDSIZE=n: This operand specifies the length (in bytes) of the index area assigned in virtual storage to the cylinder index by INDAREA. The minimum you can specify is:

n=(m+3)(keylength+6)

where

m = the number of entries to be read into virtual storage at a time.
3 = the number of dummy entries.
6 = a pointer to the cylinder.

If m is set equal to the number of prime data cylinders+1, the entire cylinder index is read into virtual storage at one time. The maximum value for n = 32767.

Bit	Cause	Explanation
0	DASD error	An uncorrectable DASD error has occurred (except wrong length record).
1	Wrong length record	A wrong length record has been detected during an I/O operation.
2	Prime area full	The next to the last track of the prime data area has been filled during the load or extension of the file. You should issue the ENDFL macro, then do a load extend on the file with nex extents given.
3	Cylinder index area full	The cylinder index area is not large enough to contain all entries needed to index each cylinder specified for the prime data area. This condition can occur during the execution of the SETFL. You must extend the upper limit of the cylinder index by using a new extent card.
4	Master index full	The master index area is not large enough to contain all the entries needed to index each track of the cylinder index. This condition can occur during SETFL. You must extend the upper limit, if you are creating the file, by using an extent card. Or, you must reorganize the file and assign a larger area.
5	Duplicate record	The record being loaded is a duplicate of the previous record.
6	Sequence check	The record being loaded is not in the sequential order required for loading.
7	Prime data area overflow	There is not enough space in the prime data area to write an EOF record. This condition can occur during the execution of the ENDFL macro.

Figure 12. FilenameC-Status or Condition Code Byte if IOROUT=LOAD

The resident index facility is suppressed if this operand is omitted, the minimum requirement is not met at assembly time, or an unrecoverable read error is encountered while reading the index.

IOAREAL=name: This operand must be included when a file is created (loaded) or when records are added to a file. It specifies the name of the output area used for loading or adding records to the file. The specified name must be the same as the name used in the DS instruction that reserves the area of storage. The ISAM routines construct the contents of this area and transfer records to DASD.

Bytes	Bits	Contents
0-3	—	DTF address
4-7	—	Virtual storage address of the record in error
8-15	—	DASD address (mbbcchhr) of the error where m is the extent sequence number and r is a record number which can be inaccurate if a read error occurred during a read of the highest level index.
16		Record identification:
	1	Data record
	2	Track index record
	3	Cylinder index record
		Master index record
		Type of operation:
	4	Not used
	5	Not used
	6	Read
	7	Write
17	—	Command code of failing CCW

Figure 13. ERREXT Parameter List

This output area must be large enough to contain the count, key, and data areas of records. Furthermore, the data-area portion must provide enough space for the sequence-link field of overflow records whenever records are added to a file (see Figure 14 on page 92).

If IOAREAL is increased to permit the reading and writing of more than one physical record on DASD at a time, the IOSIZE operand must be included when records are added to the file. In this case, the IOAREAL area must be at least as large as the number of bytes specified in the IOSIZE operand.

When simultaneously building two ISAM files using two DTFs, do not use a common IOAREAL. Also, do not use a common area for IOAREAL, IOAREAR, and IOAREAS in multiple DTFs.

IOAREAR=name: This operand must be included whenever records are processed in random order. It specifies the name of the input/output area for random retrieval (and updating). The specified name must be the same as that used in the DS instruction that reserves this area of storage.

FUNCTION	OUTPUT AREA REQUIREMENTS (IN BYTES)			
	Count	Key	Sequence Link	Data
Load Unblocked Records	8	Key Length	—	Record Length
Load Blocked Records	8	Key Length	—	Record Length x Blocking Factor
Add Unblocked Records	8	Key Length	10	Record Length
Add Blocked Records	8	Key Length	—	Record Length x Blocking Factor
	OR*			
	8	Key Length	10	Record Length
* Whichever Is Larger				

Figure 14. Output Area Requirements for Loading or Adding Records to a File by ISAM

The I/O area must be large enough to contain the data area for records. Furthermore, the data-area portion must provide enough space for the sequence-link field of overflow records (see Figure 15 on page 93).

IOAREAS=name: This operand must be included whenever records are processed in sequential order by key. It specifies the name of the input/output area used for sequential retrieval (and updating). The specified name must be the same as that used in the DS instruction that reserves this area of storage.

This I/O area must be large enough to contain the key and data areas of unblocked records and the data area for blocked records. Furthermore, the data-area portion must provide enough space for the sequence-link field of overflow records (see Figure 15 on page 93).

IOAREA2=name: This operand permits overlapping of I/O with indexed sequential processing for either the load (creation) or sequential retrieval functions. Specify the name of an I/O area to be used when loading or sequentially retrieving records. The I/O area must be at least the length of the area specified by either the IOAREAL operand for the load function or the IOAREAS operand for the sequential retrieval function. If the operand is omitted, one I/O area is assumed. If TYPEFLE=RANSEQ, this operand must not be specified.

IOREG=(r): This operand must be included whenever records are retrieved and processed directly in the I/O area. It specifies the register that ISAM uses to indicate which individual record is available for processing. ISAM puts the address of the current record in the designated register (any of 2 through 12) each time a READ, WRITE, GET, or PUT is executed.

FUNCTION	I/O AREA REQUIREMENTS (IN BYTES)			
	Count	Key	Sequence Link	Data
Retrieve Unblocked Records	—	Key Length for sequential unblocked records	10	Record Length
Retrieve Blocked Records	—	——	—	Record Length (including keys) x Blocking Factor
			OR*	
	—	——	10	Record Length
* Whichever Is Larger				

Figure 15. I/O Area Requirements for Random or Sequential Retrieval by ISAM

IOROUT={LOAD|ADD|RETRVE|ADDRTR}: This entry must be included to specify the type of function to be performed. The parameters have the following meanings:

LOAD
To build a logical file on a DASD or to extend a file beyond the highest record presently in a file.

ADD
To insert new records into a file.

RETRVE
To retrieve records from a file for either random or sequential processing and/or updating

ADDRTR
To both insert new records into a file (ADD) and retrieve records for processing and/or updating (RTR).

IOSIZE=n: This operand specifies the (decimal) number of bytes in the virtual-storage area assigned for the add function using IOAREAL. The number n can be computed using the following formula:

    n = m(keylength+blocksize+40)+24

where m is the maximum number of physical records that can be read into virtual storage at one time; 40 is the sum of 8 for the count field and 32 for an ISAM CCW; 24 is another ISAM CCW. The number n must be at least equal to

    (keylength+blocksize+74)

This formula accounts for a needed sequence link field for unblocked records or short blocks (see Figure 14 on page 92 and Figure 15).

If the operand is omitted, or if the minimum requirement is not met, no increase in throughput is realized.

The number n should not exceed the track capacity because throughput cannot be increased by specifying a number larger than the capacity of a track.

KEYARG=name: This operand must be included for random READ/WRITE operations and sequential retrieval initiated by key. It specifies the symbolic name of the key field in which you must supply the record key to ISAM.

KEYLEN=n: This operand must be included to specify the number of bytes in the record key.

KEYLOC=n: This operand must always be specified if RECFORM=FIXBLK. It supplies ISAM with the high-order position of the key field within the data record. That is, if the key is recorded in positions 21-25 of each record in the file, this operand should specify 21.

ISAM uses this specification to locate (by key) a specified record within a block. The key area of a block of records contains the key of the highest record in the block. To search for any other records, ISAM locates the proper block and then examines the key field within each record in the block.

MODNAME=name: This operand may be used to specify the name of the logic module used with the DTF table to process the file. If the logic module is assembled with the program, the MODNAME in the DTF must specify the same name as the ISMOD macro. If this entry is omitted, standard names are generated for calling the logic module. If two DTF macros call for different functions that can be handled by a single module, only one module is called.

MSTIND=YES: This operand is included whenever a master index is used or is to be built for a file. The location of the master index is specified by an EXTENT job control statement.

NRECDS=n: This operand specifies the number of logical records in a block (called the blocking factor). It is required only if RECFORM=FIXBLK. For FIXBLK, **n** must be greater than 1; for FIXUNB, **n** must be =1.

RDONLY=YES: This operand is specified if the DTF is used with a read-only module. Each time a read-only module is entered, register 13 must contain the address of a 72-byte doubleword-aligned save area. Each task should have its own uniquely defined save area. Register 13 must contain the address of the save area associated with the task each time an imperative macro (except OPEN, OPENR, LBRET, SETL, or SETFL) is issued. The fact that the save areas are unique for each task makes the module reentrant (that is, capable of being used concurrently by several tasks).

RECFORM={FIXUNB|FIXBLK}: This operand specifies whether records are blocked or unblocked. FIXUNB is used for unblocked records, and FIXBLK for blocked records. If FIXBLK is specified, the key of the highest record in the block becomes the key for the block and must be recorded in the key area.

The specification that is included when the logical file is loaded onto a DASD must also be included whenever the file is processed.

Records in the overflow area(s) are always unblocked, but this has no effect on this operand. RECFORM refers to records in the prime data area only.

RECSIZE=n: This operand must be included to specify the number of characters in the data area of each individual record. This operand should specify the same number for additions and retrieval as indicated when the file was created.

SEPASMB=YES: Include this operand only if the DTFIS is to be assembled separately. This causes a CATALR card with the filename to be punched ahead of the object deck and defines the filename as an ENTRY point in the assembly. If the operand is omitted, the assembler assumes that the DTF is assembled together with the problem program.

TYPEFLE={RANDOM|SEQNTL|RANSEQ}: This operand must be included when IOROUT=RETRVE or IOROUT=ADDRTR. The operand specifies the type(s) of processing performed by your program for the file.

RANDOM is used for random processing. Records are retrieved in random order specified by key.

SEQNTL is used for sequential processing. Your program specifies the first record retrieved, and thereafter ISAM retrieves records in sequential order by key. The first record is specified by key, ID, or the beginning of the logical file (see 'SETL Macro').

RANSEQ is used if both random and sequential processing are to be performed for the same file. If RANSEQ is specified, the IOAREA2 operand must not be specified.

TYPEFLE is not required for loading or adding functions.

VERIFY=YES: Use this operand if you want to check the parity of disk records after they are written. If this operand is omitted, any records written on a disk are not verified.

WORKL=name: This operand must be included whenever a file is created (loaded) or records are added to a file. It specifies the name of the work area in which you must supply the data records to ISAM for loading or adding to the file. The specified name must be the same as the name used in the DS instruction that reserves this area of storage.

This work area must provide space for one logical record when a file is created (for blocked records: data; for unblocked records: key and data).

The original contents of WORKL are changed due to record shifting in the ADD function.

WORKR=name: When records are processed in random order, this operand must be included if the individual records are to be processed in a work area rather than in the I/O area. It specifies the name of the work area. This name must be the same as the name used in the DS instruction that reserves this area of storage. This area must provide space for one logical record (data area). When this entry is included and a READ (or WRITE) macro is executed, ISAM moves the individual record to (or from) this area.

WORKS=YES: When records are processed in sequential order, this operand must be included if the individual records are processed in work areas rather than in the I/O area. Each GET and PUT macro must specify the name of the work area to or from which ISAM is to move the record. When processing unblocked records, the area must be large enough for one record (data area) and the record key (key area). For blocked records, the area must be large enough for one logical record (data area) only. The work area requirements are as shown in Figure 16 .

	Unblocked Records	Blocked Records
Load	(KL+DL) or 10*	DL or 10*
ADD	(KL+DL) or 10*	DL or (KL+10)*
Random Retrieve	DL	DL
Sequential Retrieve	KL+DL	DL
K = Key D = Data L = Length		
* whichever is greater		

Figure 16. Work Area Requirements

DTFMR

```
--------------------------------------------------
Name          Operation     Operand
--------------------------------------------------
[name]        DTFMR         DEVADDR=SYSxxx
                            ,IOAREA1=name
                            [,ADDAREA=n]
                            [,ADDRESS=DUAL]
                            [,BUFFERS={25|n}]
                            [,ERROPT=name]
                            [,EXTADDR=name]
                            [,IOREG=(r)
                            [,MODNAME=name]
                            [,RECSIZE={80|n}]
                            [,SECADDR=SYSnnn]
                            [,SEPASMB=YES]
                            [,SORTMDE={ON|OFF}]
--------------------------------------------------
```

DTFMR defines an input file processed on a 1255, 1259, or 1419 magnetic character reader, or a 1270 or 1275 optical character reader/sorter.

ADDAREA=n: This operand must be included only if an additional buffer work area is needed. The parameter n specifies the number of additional bytes you desire in each buffer. The sum of the ADDAREA and RECSIZE specifications must not exceed 250. This area can be used as a work area and/or output area and is reset to binary zeros when the next GET or READ for the file is executed.

ADDRESS=DUAL: This operand must be included only if the 1419 or 1275 contains the dual address adapter. If the single address adapter is used, this operand must be omitted.

BUFFERS={25|n}: This operand is included to specify the number of buffers in the document buffer area. The limits for n are 12 and 254. 25 is assumed if this operand is omitted.

DEVADDR=SYSxxx: This operand is required and specifies the symbolic unit to be associated with the file. The symbolic unit represents an actual I/O device address used in the ASSGN job control statement to assign the actual I/O device address to the file.

ERROPT=name: This operand may be included only if the CHECK macro is used. The name parameter specifies the name of the routine that the CHECK macro branches to if any error condition is posted in byte 0, bits 2 to 4 (and bit 5, if no control address is specified in the CHECK macro) of the buffer status indicators. It is your responsibility to exit from this routine (see the 'CHECK Macro'.)

EXTADDR=name: This operand specifies the name of your stacker selection routine to which control is given when an external interrupt is encountered while reading and sorting the documents internally. This operand may be omitted only when you specify SORTMDE=OFF.

IOAREA1=name: This operand is required and specifies the name of the document buffer area that will be used by the file. Figure 4 on page 31 shows the format of the document buffer area.

IOREG={(2)|(r)}: This operand specifies the general-purpose register (one of 2 to 12) that the IOCS routines and your routines use to indicate which individual document buffer is available for processing. IOCS puts the address of the current document buffer in the specified register each time a GET or READ is issued. Register 2 is assumed if this operand is omitted.

The same register may be specified in the IOREG entry for two or more files in the same program, if desired. In this case, your program may need to store the address supplied by IOCS for each record.

MODNAME=name: This operand specifies the name of the logic module generated by MRMOD. If the operand is omitted, IOCS generates the standard system module name.

RECSIZE={80|n}: This operand specifies the actual length of the data portion of the buffer. The record size specified must be the size of the largest record processed. If this operand is omitted, a record size of 80 is assumed. The sum of the ADDAREA and RECSIZE specifications must not exceed 250.

SECADDR=SYSnnn: This operand specifies the symbolic unit to be associated with the secondary control unit address if the 1419 or 1275 with the dual address adapter and LITE macro are utilized. The operand should be omitted if the pocket LITE macro is not being used.

SEPASMB=YES: Include this operand only if the DTFMR is to be assembled separately. This causes a CATALR card with the filename to be punched ahead of the object deck and defines the filename as an ENTRY point in the assembly. If the operand is omitted, the assembler assumes that the DTF is assembled together with the problem program.

SORTMDE={ON|OFF}: This operand specifies the method of sorting done on the 1419. SORTMDE=ON indicates that the program sort mode is being used. SORTMDE=OFF indicates that sorting is under control of the magnetic character reader. If the operand is omitted, the program sort mode is assumed.

DTFMT

```
----------------------------------------------------------
Name         Operation    Operand
----------------------------------------------------------
[name]       DTFMT        BLKSIZE=n
                          ,DEVADDR=SYSxxx
                          ,EOFADDR=name
                          ,FILABL={NO|STD|NSTD}
                          ,IOAREA1=name
                          [,ASCII=YES]
                          [,BUFOFF=n]
                          [,CKPTREC=YES]
                          [,ERREXT=YES]
                          [,ERROPT={IGNORE|SKIP|name}]
                          [,HDRINFO=YES]
                          [,IOAREA2=name]
                          [,IOREG=(r)]
                          [,LABADDR=name]
                          [,LENCHK=YES]
                          [,MODNAME=name]
                          [,NOTEPNT={YES|POINTS}]
                          [,RDONLY=YES]
                          [,READ={FORWARD|BACK}]
                          [,RECFORM=xxxxxx]
                          [,RECSIZE={n|(r)}]
                          [,REWIND={UNLOAD|NORWD}]
                          [,SEPASMB=YES]
                          [,TPMARK={YES|NO}]
                          [,TYPEFLE={INPUT|OUTPUT|WORK}]
                          [,VARBLD=(r)]
                          [,WLRERR=name]
                          [,WORKA=YES]
----------------------------------------------------------
```

The DTFMT macro defines a magnetic tape file.

If not otherwise stated, the operands of the DTFMT macro can be specified for all three types of files (input, output, or work).

ASCII=YES: This operand specifies that processing of ASCII tapes is required (see Appendix B). If this operand is omitted, EBCDIC processing is assumed. ASCII=YES is not permitted for work files.

BLKSIZE=n: Enter the length of the I/O area. If the record format is variable or undefined, enter the length of the largest block of records. If a READ or WRITE macro specifies a length greater than n for work files, the record to be read or written will be truncated to fit in the I/O area. The maximum block size is 32,767 bytes. The minimum size of a physical tape record (gap to gap) is 12 bytes. A record of eleven bytes or less is treated as noise.

For output processing of variable records, the minimum physical record length is 18 bytes. If less than 18 bytes are specified for variable blocked or variable unblocked records, BLKSIZE=18 is assumed.

For output processing of spanned records, the minimum physical record length is 18 bytes. If SPNBLK or SPNUNB and TYPEFLE=OUTPUT are specified in the DTFMT and the BLKSIZE is invalid or less than 18 bytes, an MNOTE is generated and BLKSIZE=18 is assumed.

For ASCII tapes, the BLKSIZE includes the length of any block prefix or padding characters present. If ASCII=YES and BLKSIZE is less than 18 bytes (for fixed-length records only) or greater than 2048 bytes, an MNOTE is generated because this length violates the limits specified by American National Standards Institute, Inc.

BUFOFF={0|n}: For ASCII tapes, this operand indicates the length of the block prefix. Enter the length of the block prefix if processing of the block prefix is required. This operand can only be included when ASCII=YES is specified; it is not allowed for work files. The contents of this field are not passed on to you.

n can have the following values:

Value	**Condition**
0-99	If TYPEFLE=INPUT
0	IF TYPEFLE=OUTPUT
4	If TYPEFLE=OUTPUT and RECFORM=VARUNB or VARBLK. In this case, the program automatically inserts the physical record length in the block prefix.

CKPTREC=YES: This operand is necessary if an input tape has checkpoint records interspersed among the data records. IOCS bypasses any checkpoint records encountered. This operand must not be included when ASCII=YES.

DEVADDR={SYSRDR|SYSIPT|SYSPCH|SYSnnn|SYSLST}: This operand specifies the symbolic unit to be associated with the file. An ASSGN job control statement assigns an actual channel and unit number to the unit. The ASSGN job control statement contains the same symbolic name as DEVADDR. When processing ASCII tapes, you must specify a programmer logical unit (SYSnnn).

EOFADDR=name: This operand specifies the name of your end-of-file routine. IOCS automatically branches to this routine on an end-of-file condition. This entry must be specified for input and work files.

In your routine, you can perform any operations required for the end of file (generally you issue the CLOSE macro for the file). IOCS detects end-of-file conditions in magnetic tape input by reading a tapemark and EOF when standard labels are specified. If standard labels are not specified, IOCS assumes an end-of-file condition when the tapemark is read, or, if the unit is assigned to SYSRDR or SYSIPT, when a /* is read. You must determine, in your routine, that this actually is the end of the file.

ERREXT=YES: This operand enables your ERROPT or WLRERR routine to return to IOCS by means of the ERET (error return) macro. It also enables nonrecoverable I/O errors other than tape read data checks to be indicated to your program. If ERREXT=YES is specified, the ERROPT=name operand must also be specified.

ERROPT={IGNORE|SKIP|name}: This operand specifies functions to be performed when a tape read data check or (when ERREXT=YES is specified) a tape write check (non-recoverable I/O error) is encountered. Either IGNORE, SKIP, or the symbolic name of an error routine can be specified. The functions of these specifications are:

IGNORE

The error condition is completely ignored, and the records are made available for processing. When reading spanned records, the entire spanned record or a block of spanned records is returned to the user rather than just the one physical record in which the error occurred.

On output, the error is ignored and the physical record containing the error is treated as a valid record. The remainder, if any, of the spanned record segments are written, if possible.

SKIP

No records in the error block are made available for processing. The next block is read from tape, and processing continues with the first record of that block. The error block is included in the block count. When reading spanned records, the entire spanned record or a block of spanned records is skipped rather than just one physical record.

On output, the error is ignored and the physical record containing the error is treated as a valid record. The remainder, if any, of the spanned record segments are written.

name

IOCS branches to your error routine named by this parameter

- only when a tape read data check is encountered (ERREXT=YES <u>not</u> specified), or
- when an unrecoverable I/O error is encountered (ERREXT=YES specified).

In your error routine, you can process or make note of the error condition as desired.

The ERROPT operand applies to wrong-length records if the WLRERR operand is not included. If both ERROPT and WLRERR are omitted and wrong-length records occur, IOCS assumes the IGNORE option.

> **Note:** For ASCII tapes, the pointer to the block in error indicates the first logical record following the block prefix.

FILABL={NO|STD|NSTD}: This operand specifies what type of labels are to be processed. STD indicates standard labels, NO indicates no labels, and NSTD indicates nonstandard labels. You must furnish a routine to check or create the nonstandard labels by using your own I/O area and an EXCP macro to read or write the labels. The entry point of this routine is the operand of LABADDR.

The specification FILABL=NSTD is not permitted for ASCII files (that is, when ASCII=YES). Labels and tape data are assumed to be in the same mode.

HDRINFO=YES: This operand, if specified with FILABL=STD, causes IOCS to print standard header label information (fields 3-10) on SYSLOG each time a file with standard labels is opened. It also prints the filename, logical unit, and device address each time an end-of-volume condition is detected. Both FILABL=STD and HDRINFO=YES must be specified for header label information to be printed.

IOAREA1=name: This operand specifies the name of the I/O area. When variable-length records are processed, the size of the I/O area must include four bytes for the block size. This operand does not apply to work files.

IOAREA2=name: This operand specifies the name of a second I/O area. When variable-length records are processed, the size of the I/O area must include four bytes for the blocksize. This operand does not apply to work files.

IOREG=(r): This operand specifies the register in which IOCS places the address of the logical record that is available for processing if:

- two input or output areas are used.
- blocked input or output records are processed in the I/O area.
- variable unblocked records are read.
- undefined records are read backwards.
- neither BUFOFF=0 nor WORKA=YES is specified for ASCII files.

For output files, IOCS places, in the specified register, the address of the area where you can build a record. Any of registers 2 to 12 may be specified.

This operand cannot be used if WORKA=YES.

LABADDR=name: Enter the symbolic name of your routine to process user-standard or nonstandard labels. For ASCII tapes, this operand may be used only for writing and checking user standard labels that conform to American National Standards Institute, Inc., standards. You must process these labels in EBCDIC. Non-standard user labels are not permitted. This operand does not apply to work files.

LENCHK=YES: This operand applies only to ASCII tape input if BUFOFF=4 and RECFORM=VARUNB or VARBLK. It must be included if the block length (specified in the block prefix) is to be checked against the physical record length. If the two lengths do not match, the action taken is the same as described under the WLRERR operand, but the WLR bit (byte 5, bit 1) in the DTF is not set.

MODNAME=name: This operand specifies the name of the logic module used with the DTF table to process the file. If the logic module is assembled with the program, this operand must specify the same name as the MODNAME operand of the MTMOD macro. If this operand is omitted, standard names are generated for calling the logic module. If two DTF macros call for different functions that can be handled by a single module, only one module is called. For example, if one DTF specifies READ=FORWARD and another specifies READ=BACK, only one logic module capable of handling both functions is called.

NOTEPNT={POINTS|YES}: If the parameter YES is specified, the NOTE, POINTW, POINTR, or POINTS macros can be issued for a tape work file. If POINTS is specified, only POINTS macros can be issued for tape work files. The NOTEPNT operand must not be specified for ASCII tape files because ASCII work files are not supported.

RDONLY=YES: This operand is specified if the DTF is used with a read-only module.

READ={FORWARD|BACK}: This operand specifies, for input and work files, the direction in which the tape is read. If READ=BACK is specified and a wrong-length record smaller than the I/O area is encountered, the record is read into the I/O area right-justified.

RECFORM={FIXUNB|FIXBLK|VARUNB|VARBLK|SPNBLK|SPNUNB|UNDEF}: This operand specifies the type of EBCDIC or ASCII records in the input or output file. Enter one of the following parameters:

FIXUNB	For fixed-length unblocked records (default)
FIXBLK	For fixed-length blocked records
VARUNB	For variable-length unblocked records
VARBLK	For variable-length blocked records
SPNBLK	For spanned variable-length blocked records (EBCDIC only)

SPNUNB	For spanned variable-length unblocked records (EBCDIC only)
UNDEF	For undefined records

Work files may use only FIXUNB or UNDEF.

RECSIZE={n|(r)}: For fixed-length blocked records, RECSIZE is required. It specifies the number of characters in each record.

When processing spanned records, you must specify RECSIZE=(r) where r is a register that contains the length of each record.

For undefined records, this entry is required for output files but is optional for input files. It specifies a general register (any of 2 to 12) that contains the length of the record. On output, you must load the length of each record into the register before you issue a PUT macro.

Spanned-record output requires a minimum record length of 18 bytes. A physical record less than 18 bytes is padded with binary zeros to complete the 18-byte requirement. This applies to both blocked and unblocked records. If specified for input, IOCS provides the length of the record transferred to virtual storage. This operand does not apply to work files.

REWIND={UNLOAD|NORWD}: If this specification is not included, tapes are automatically rewound to load point, but not unloaded, on an OPEN or OPENR or a CLOSE or CLOSER macro or on an end-of-volume condition. If other operations are desired for a tape input or output file, specify:

UNLOAD
to rewind the tape on an OPEN and to rewind and unload on a CLOSE or on an end-of-volume condition.

NORWD
to prevent rewinding the tape at any time. This option positions the read/write head between the two tapemarks that indicate the end-of-file condition.

SEPASMB=YES: Include this operand only if the DTFMT is to be assembled separately. This causes a CATALR card with the filename to be punched ahead of the object deck and the filename to be defined as an ENTRY point in the assembly. If the operand is omitted, the assembler assumes that the DTF is assembled together with the problem program.

TPMARK={YES|NO}: A tapemark is normally written for an output file if nonstandard labels are specified (FILABL=NSTD). If no tapemark is desired, TPMARK=NO should be specified. If TPMARK=NO is specified together with FILABL=STD, the former specification is ignored. If FILABL=NO is specified or the FILABL operand is omitted, TPMARK=YES must be specified for IOCS to write a tapemark ahead of the first data record.

TYPEFLE={INPUT|OUTPUT|WORK}: Use this operand to indicate whether the file is used for input or output. If INPUT is specified, the GET macro is used. If OUTPUT is specified, the PUT macro is used. If WORK is specified, the READ/WRITE, NOTE/POINTx, and CHECK macros are used.

The specification of WORK in this operand is not permitted for ASCII files.

VARBLD=(r): This entry is required whenever variable-length blocked records are built directly in the output area (no work area is specified). It specifies the number (r) of a general-purpose register (any of 2 to 12) that always contains the length of the available space remaining in the output area.

IOCS calculates the space still available in the output area, and supplies it to you in the VARBLD register after the PUT macro is issued for a variable-length record. You can then compare the length of the next variable-length record with the available space to determine whether the record will fit in the remaining area. This check must be made before the record is built. If the record does not fit, issue a TRUNC macro to transfer the completed block of records to the tape. The current record is then built as the first record of the next block.

WLRERR=name: This operand applies only to tape input files. It specifies the name of your routine to receive control if a wrong-length record is read. If the WLRERR entry is omitted but a wrong-length record is detected by IOCS, one of the following conditions results:

- If the ERROPT entry is included for this file, the wrong-length record is treated as an error block, and handled according to your specifications for an error (IGNORE, SKIP, or name of error routine).
- If the ERROPT entry is not included, IOCS assumes the IGNORE option of ERROPT.

WORKA=YES: If I/O records are processed in work areas instead of in the I/O areas, specify this operand. You must set up the work areas in virtual storage. The symbolic address of the work area, or a general-purpose register containing the address, must be specified in each GET or PUT. Omit IOREG if this operand is included. WORKA=YES is required for spanned record processing. It does not apply to work files.

DTFOR

```
-------------------------------------------------------------
Name          Operation    Operand
-------------------------------------------------------------
[name]        DTFOR        COREXIT=name
                           ,DEVADDR=SYSxxx
                           ,EOFADDR=name
                           ,IOAREA1=name
                           [,BLKFAC=n]
                           [,BLKSIZE=n]
                           [,CONTROL=YES]
                           [,DEVICE=xxxxx]
                           [,HEADER=YES]
                           [,HPRMTY=YES]
                           [,IOAREA2=name]
                           [,IOREG=(r)]
                           [,MODNAME=name]
                           [,RECFORM={FIXUNB|FIXBLK|UNDEF}]
                           [,RECSIZE=(n)]
                           [,SEPASMB=YES]
                           [,WORKA=YES]
-------------------------------------------------------------
```

This macro is used to define an input file to be processed on a 1287 optical reader or 1288 optical page reader. The macro is not used for the 3881 optical mark reader; for the 3881, use the DTFCD macro.

If not otherwise stated, the operands of the DTFOR macro can be specified for all three types of devices (1287T, 1287D, and 1288).

BLKFAC=n: For the 1287T, undefined journal tape records are processed with greater throughput speeds when this operand is included. This is accomplished by reading groups of lines as blocked records. When undefined records are processed, BLKFAC specifies the blocking factor (n) that determines the number of lines read (through CCW chaining) as a block of data by one physical read. Deblocking is accomplished automatically by IOCS when the GET macro is used. The BLKFAC parameter is not used with RECFORM=FIXBLK, because the blocking factor is determined from the BLKSIZE and RECSIZE parameters. If the operand is included for FIXBLK, FIXUNB, or document processing, the operand is noted (in an MNOTE) and ignored.

BLKSIZE={38|n} This operand indicates the size of the input area specified by IOAREA1. 38 is the default. For journal tape processing, BLKSIZE specifies the maximum number of characters that can be transferred to the area at any one time.

When undefined journal tape records are read, the area must be large enough to accommodate the longest record to be read if the BLKFAC parameter is not specified. If the BLKFAC parameter is specified, the BLKSIZE value must be determined by multiplying the maximum length that must be accommodated for an undefined record by the blocking factor desired. A BLKSIZE value smaller than this results in truncated data.

If two input areas are used for journal tape processing (IOAREA1 and IOAREA2), the size specified in this entry is the size of each I/O area.

CONTROL=YES: This entry must be included if a CNTRL macro is issued for a file. A CNTRL macro issues orders to the optical reader to perform nondata operations such as line marking, stacker selecting, document incrementing, etc.

COREXIT=name: COREXIT provides an exit to your error correction routine for the 1287 or 1288. After a GET, WAITF, or CNTRL macro is executed (to increment or eject and/or stacker select a document), an error condition causes an error correction routine to be entered with an error indication provided in filename+80. The byte at filename+80 contains the following codes indicating the conditions that occurred during the last line or field read. The byte should also be tested after issuing the optical reader macros DSPLY, RESCN, RDLNE, CNTRL READKB, and CNTRL MARK. More than one error condition may be present.

Code		
Dec	**Hex**	**Meaning**
1	X'01'	A data check has occurred. Five read attempts for journal tape processing or three read attempts for document processing were made.
2	X'02'	The operator corrected one or more characters from the keyboard (1287T) or a hopper empty condition (see HPRMTY=YES operand) has occurred (1287D).
4	X'04'	A wrong-length record condition has occurred (for journal tapes, five read attempts were made; for documents, three read attempts were made). Not applicable for undefined records.
8	X'08'	An equipment check resulted in an incomplete read (ten read attempts were made for journal tapes or three for documents). If an equipment check occurs on the first character in the record, when processing undefined journal tape records, the RECSIZE register contains zero, and the IOREG (if used) points to the rightmost position of the record in the I/O area. You should test the RECSIZE register before moving records from the work area or the I/O area.
16	X'10'	A nonrecoverable error occurred.

32	X'20'	For the 1288, reading in unformatted mode, the end-of-page (EOP) condition has been detected. Normally, on an EOP indication, the problem program ejects and stacker selects the document. After issuing one of the macros CNTRL ESD, CNTRL SSD, CNTRL EJD in your COREXIT routine, a late stacker selection condition occurred. For the 1287, a stacker select was given after the allotted elapsed time and the document was put in the reject pocket.
64	X'40'	The 1287D scanner was unable to locate the reference mark (for journal tapes, ten read attempts were made; for documents, three read attempts were made).

The byte filename+80 can be interrogated to determine the reason for entering the error correction routine. Choice of action in your error correction routine is determined by the particular application.

If you issue I/O macros to any device other than the 1287 and/or 1288 in the COREXIT routine, you must save registers 0, 1, 14, and 15 upon entering the routine, and restore these registers before exiting. Furthermore, if I/O macros (other than the GET, WAITF, and/or READ, which cannot be used in COREXIT) are issued to the 1287 and/or 1288 in this routine, you must also save and later restore registers 14 and 15 before exiting. All exits from COREXIT should be to the address specified in register 14. This provides a return to the point from which the branch to COREXIT occurred. If the command chain bit is on in the READ CCW for which the error occurred, IOCS completes the chain upon return from the COREXIT routine.

Note: Do not issue a GET, READ, OPEN, or WAITF macro to the 1287 or 1288 in the error correction routine. Do not process records in the error correction routine. The record that caused the exit to the error routine is available for processing upon return to the mainline program. Any processing included in the error routine would be duplicated after return to the mainline program.

When processing journal tapes, a nonrecovery error (torn tape, tape jam, etc.) normally requires that the tape be completely reprocessed. In this case, your routine must not branch to the address in register 14 from the COREXIT routine or a program loop will occur. Following an unrecoverable error:

- the optical reader file must be closed.
- the condition causing the nonrecovery must be cleared.
- the file must be reopened before processing can continue.

If a nonrecoverable error occurs while processing documents (indicating that a jam occurred during a document incrementation operation, or a scanner control failure has occurred, or an end-of-page condition, etc.), the document should be removed either manually or by nonprocess runout. In such cases, your program should branch to read the next document.

If the 1287 or 1288 scanner is unable to locate the document reference mark, the document cannot be processed. In this case, the document must be ejected and stacker selected before attempting to read the following document or a program loop will result.

Whenever a nonrecoverable error occurs, your COREXIT routine must not branch to the address in register 14 to return to IOCS. Instead, the routine should ignore any output resulting from the document.

Eight binary error counters are used to accumulate totals of certain 1287 and 1288 error conditions. Each of these counters occupies four bytes, starting at filename+48. Filename is the name specified in the DTF header entry. The error counters are:

Counter and Address	**Contents**
1 filename+48	Equipment check (see Note, below).
2 filename+52	Equipment check uncorrectable after ten read attempts for journal tapes or three read attempts for documents (see Note, below).
3 filename+56	Wrong-length records (not applicable for undefined records).
4 filename+60	Wrong-length records uncorrectable after five read attempts for journal tapes or three read attempts for documents (not applicable for undefined records).
5 filename+64	Keyboard corrections (journal tape only).
6 filename+68	Journal tape lines (including retried lines) or document fields (including retried fields) in which data checks are present.
7 filename+72	Lines marked (journal tape only).
8 filename+76	Count of total lines read from journal tape or the number of CCW chains executing during document processing.

Note: Counters 1 and 2 apply to equipment checks that result from incomplete reads or from the inability of the 1287 or 1288 scanner to locate a reference mark (when processing documents only).

All the previous counters contain binary zeros at the start of each job step. You may list the contents of these counters for analysis at end of file, or at end of job, or you may ignore the counters. The binary contents of the counters should be converted to a printable format.

DEVADDR=SYSnnn: This operand specifies the logical unit (SYSnnn) to be associated with the file. The logical unit represents an actual I/O device address used in the ASSGN job control statement to assign the actual I/O device address to this file.

DEVICE={1287D|1287T}: This operand specifies the I/O device associated with this file. 1287D specifies a 1287 or 1288 document file. 1287T specifies a 1287 journal tape file.

From this specification, IOCS sets up the device-dependent routines for this file. For document processing you must code the CCWs.

If this operand is omitted, 1287D is assumed.

EOFADDR=name: This operand specifies the name of your end-of-file routine. IOCS automatically branches to this routine on an end-of-file condition.

When reading data from documents, you can recognize an end-of-file condition by pressing the end-of-file key on the console when the hopper is empty. When processing journal tapes on a 1287, you can detect an end-of-file by pressing the end-of-file key after the end of the tape is sensed.

When IOCS detects an end-of-file condition, it branches to your routine specified by EOFADDR. You must determine whether the current roll is the last roll to be processed when handling journal tapes. Regardless of the situation, the tape file must be closed for each roll within your EOF routine. If the current roll is not the last, OPEN must be issued. The OPEN macro allows header (identifying) information to be entered at the reader keyboard and read by the processor when using logical IOCS.

The same procedure can be used for 1287 processing of multiple journal tape rolls, as well as the method described under 'OPEN Macro' in the section 'Imperative Macros'.

HEADER=YES: This operand cannot be used for 1288 files. This operand is required if the operator is to key in header (identifying) information from the 1287 keyboard. The OPEN routine reads the header information only when this entry is present. If the entry is not included, OPEN assumes no header information is to be read. The header record size can be as large as the BLKSIZE entry and is read into the high-order positions of IOAREA1.

HPRMTY=YES: This operand is included (for the 1287D or 1288) if you want to be informed of the hopper empty condition. This condition occurs when a READ is issued and no document is present, and is recognized at WAITF time. When a hopper empty condition is detected, your COREXIT routine is entered with X'02' stored in filename+80.

This operand should be used when processing documents in the time-dependent mode of operation, which allows complete overlapping of processing with reading. See the appropriate IBM 1287 device manuals for processing details. With this method of processing, specifying HPRMTY=YES allows you to check for a hopper empty condition in your COREXIT routine. You can then select into the proper hopper the previously ejected document before return from COREXIT (via register 14).

IOAREA1=name: This operand is included to specify the name of the input area used by the file. When opening a file and before each journal tape input operation to this area, the designated area is set to binary zeros and the input routines then transfer records to this area. For document processing, the area is cleared only when the file is opened.

IOAREA2=name: A second input area can be allotted only for a journal tape file (on a 1287T). This permits an overlap of data transfer and processing operations. The specified second I/O area is set to binary zeros before each input operation to this area occurs.

IOREG={(2)|(r)}: This operand specifies a general-purpose register (any one of 2 to 12) that the input routines use to indicate the beginning of records for a journal tape file (on a 1287T). The same register may be specified in the IOREG operand for two or more files in the same program, if desired. In this case, your program may need to store the address supplied by IOCS for each record. Whenever this entry is included for a file, the DTFOR entry WORKA must be omitted, and the GET macro must not specify a work area.

A read by an optical reader is accomplished by a backward scan. This places the rightmost character in the record into the rightmost position of the I/O area and subsequent characters in sequence from right to left. The register defined by IOREG indicates the leftmost position of the record.

MODNAME=name: This operand may be used to specify the name of the logic module used with the DTF table to process the file. If the logic module (ORMOD) is assembled with the program, the MODNAME parameter in this DTF must specify the same name as the ORMOD macro.

If this entry is omitted, standard names are generated for calling the logic module. If two different DTF macros call for different functions that can be handled by a single module, only one standard-named module is called.

RECFORM={FIXUNB|FIXBLK|UNDEF}: This operand specifies the type of records in an optical reader file. One of the following may be specified:

FIXUNB	For fixed-length unblocked records (default).
FIXBLK	For fixed-blocked records in journal tape mode.
UNDEF	For undefined records.

RECSIZE={n|[(3)|(r)]}: For fixed-length unblocked records, this operand should be omitted and no register is assumed.

For fixed-length blocked records (journal tape mode), this operand must be included to specify the number, n, of characters in an individual record. The input routines use this number to deblock records, and to check the length of input records. If this operand is omitted, an MNOTE is flagged in the macro assembly and fixed-length unblocked records are assumed.

For undefined journal tape records, this entry specifies the number (r) of the general-purpose register in which IOCS provides the length of each input record. For undefined document records, RECSIZE contains only the length of the last field of a document read by the CCW chain that you supply. Any one of registers 2 through 12 may be specified, but if the operand is omitted, register 3 is assumed.

> **Note:** When processing undefined records in document mode, you gain complete usage of the register normally used in the RECSIZE operand. You can do this by ensuring that the suppress-length-indication (SLI) flag is always on when processing undefined records.

SEPASMB=YES: Include this operand only if the DTFOR is to be assembled separately. This causes a CATALR card with the filename to be punched ahead of the object deck and the filename to be defined as an ENTRY point in the assembly. If the operand is omitted, the assembler assumes that the DTF is assembled together with the problem program.

WORKA=YES: Input records (journal tape on a 1287T only) can be processed in work areas instead of in the input areas. If this is planned, the operand WORKA=YES must be specified, and you must set up the work area in storage. The symbolic name of the work area, or a general-purpose register containing the address of the work area, must be specified in each GET macro. When GET is issued, IOCS left-justifies the record in the specified work area. Whenever this operand is included for a file, the DTFOR IOREG operand must be omitted.

DTFPH

```
-----------------------------------------------------------
Name        Operation   Operand
-----------------------------------------------------------
[name]      DTFPH       TYPEFLE={INPUT|OUTPUT}
                        [,ASCII=YES]
                        [,CISIZE=n]
                        [,CCWADDR=name]
                        [,DEVADDR=SYSxxx]
                        [,DEVICE=xxxx]
                        [,HDRINFO=YES]
                        [,LABADDR=name]
                        [,MOUNTED={ALL|SINGLE}]
                        [,XTNTXIT=name]
-----------------------------------------------------------
```

When physical IOCS macros (EXCP, WAIT, etc.) are used in a program, DASD, diskette, or tape files with standard labels need to be defined by the DTFPH macro (DTFxx macro for a file handled by physical IOCS). DTFPH must also be used for a checkpoint file on a disk.

Figure 17 on page 114 shows which of the DTFPH entries can or must be coded to define a checkpoint file on disk.

ASCII=YES: This operand is required to process ASCII tape files (see Appendix B). If this operand is omitted, EBCDIC processing is assumed.

CCWADDR=name: This operand allows you to use the CCB generated within the first 16 bytes of the DTFPH table. CCWADDR specifies the symbolic name of the first CCW used with the CCB generated within the DTFPH macro. This name must be the same as the name specified in the assembler CCW statement that constructs the CCW.

If this operand is omitted, the location counter value of the CCB-CCW table address constant is substituted for the CCW address.

CISIZE=n: This operand specifies the FBA Control Interval size. The value n must be an integral multiple of the FBA physical block size and, if greater than 8K, must be a multiple of 2K. The maximum value is 32768 (32K) except when assigned to SYSLST or SYSPCH, when the maximum is 30720 (30K).

If CISIZE is omitted, CISIZE=0 is assumed. For FBA devices, control interval size may be overridden for an output file at execution time by specifying the CISIZE parameter of the DLBL job control statement. For an input file, the CISIZE value in the format-1 label is used.

DEVADDR=SYSxxx: This operand must specify the symbolic unit (SYSxxx) associated with the file if a symbolic unit is not provided via an EXTENT job control statement. If a symbolic unit is provided,

Operand	Optional	Required
CCWADDR=name	x	
CISIZE=n	x	
DEVADDR=SYSnnn	x	
DEVICE=2311, 2314, 3330, 3340, 3350, DISK		x
LABADDR=name	x	
MOUNTED=SINGLE		x
TYPEFLE=OUTPUT		x

Figure 17. Operands to Define a Checkpoint File on Disk

its specification overrides a DEVADDR specification. This specification, or symbolic unit, represents an actual I/O address, and is used in the ASSGN job control statement to assign the actual I/O device address to this file.

If SYSLST or SYSPCH are used as output tape units and alternate tape switching is desired upon detecting a reflective spot, the SEOV macro must be used (see 'SEOV Macro'). When processing ASCII tape files, the only valid specification is a programmer logical unit (that is, SYSnnn).

DEVICE={TAPE|DISK|2311|2314|3330|3340|3350|3540}: TAPE is the default, so if the file is contained on DASD or diskette, enter the proper identification.

TAPE applies to 8809 and any 2400/3400-series tape unit, and is the only valid entry in this operand for ASCII files.

DISK is a general DASD device specification including CKD and FBA devices. When specified, a specific device type is not required. The actual DASD device is determined when the file is opened.

HDRINFO=YES: This operand causes IOCS to print standard header label information (fields 3-10) on SYSLOG each time a file with standard labels is opened. Likewise, the filename, symbolic unit, and device address are printed each time an end-of-volume condition is detected. If HDRINFO=YES is omitted, no header or end-of-volume information is printed.

LABADDR=name: This operand does not apply to diskette input/output units.

You may require one or more DASD or tape labels in addition to the standard file labels. If so, you must include your own routine to check (on input) or build (on output) your label(s). Specify the symbolic name of your routine in this operand. IOCS branches to this routine after the standard label is processed.

LABADDR may be included to specify a routine for your header or trailer labels as follows:

- DASD input or output: header labels only.
- Tape input or output: header and trailer labels.

Thus, if LABADDR is specified, your header labels can be processed for an input/output DASD or tape file, and your trailer labels can be built for a tape output file. Physical IOCS reads input labels and makes them available to you for checking, and writes output labels after they are built. This is similar to the functions performed by logical IOCS.

If physical IOCS macros are used for a tape file, an OPEN must be issued for the new volume. This causes IOCS to check the HDR1 label and provides for your checking of user standard labels, if any.

When physical IOCS macros are used and DTFPH is specified for standard tape label processing, FEOV must not be issued for an input file.

MOUNTED={ALL|SINGLE}: This operand does not apply to diskette input/output units.

This operand must be included to specify how many extents (areas) of the file are available for processing when the file is initially opened. This operand must not be specified for tape.

ALL is specified if all extents are available for processing. When a file is opened, IOCS checks all labels on each disk pack and makes available all extents specified by your control statements. Only one OPEN is required for the file. ALL should be specified whenever you plan to process records in a manner similar to the direct access method.

After an OPEN is performed, you must be aware that the symbolic unit address of the first volume containing the file is in bytes 30 and 31 of the DTFPH table rather than in the CCB. Therefore, place this symbolic address into bytes 6 and 7 of the associated CCB before you issue an EXCP against this CCB in your program.

SINGLE is specified if only the first extent on the first volume is available for processing. SINGLE should be specified when you plan to process records in sequential order. IOCS checks the labels on the first pack and makes the first extent specified by your control statements available for processing. You must keep track of the extents and issue a subsequent OPEN whenever another extent is required for processing. You will find the information in the DTFPH table helpful in keeping track of the extents. The contents of the DTFPH is shown in Figure 18 on page 116 .

Bytes	Contents
0–15	CCB (symbolic unit has been initialized in the CCB).
54–57	Extent upper limits (cchh).
58–59	Seek address. For a disk it must be zero.
60–63	Extent lower limit (cchh for CKD). For FBA devices, the extent upper limit is the first physical block number of the last CI. If the number of blocks per CI is greater than 1, the extent upper limit could differ between the format–1 label and the DTF entry.

Figure 18. DTFPH Table

On each OPEN after the first, IOCS makes available the next extent specified by the control cards. When you issue a CLOSE for an output file, the volume on which you are currently writing records is indicated, in the file label, as the last volume for the file.

TYPEFLE={INPUT|OUTPUT}: This operand must be included to specify the type of file: input or output.

XTNTXIT=name: This operand does not apply to diskette input/output units.

This entry is included if you want to process label extent information. It specifies the symbolic name of your extent routine. The DTFPH operand MOUNTED=ALL must also be specified for the file.

Whenever XTNTXIT is included, IOCS branches to your routine during the initial OPEN for the file. It branches after each specified extent is completely checked and after conflicts, if any, have been resolved.

When your routine receives control, register 1 contains the address of a 14-byte area from which you can retrieve label extent information (in binary form). The layout of this area is shown in Figure 19 on page 117 .

Return to IOCS by using the LBRET macro.

Bytes	Contents
0	Extent type code.
1	Extent sequence number.
2-5	Lower limit of the extent.
6-9	Upper limit of the extent.
10-11	Symbolic unit.
12	Contains zero.
13	Not used.

Figure 19. Layout of XTNTXIT Information Area

```
-----------------------------------------------------------
Name        Operation    Operand
-----------------------------------------------------------
[name]      DTFPR        DEVADDR=SYSxxx
                         ,IOAREA1=name
                         [,ASOCFLE=filename]
                         [,BLKSIZE=n]
                         [,CONTROL=YES]
                         [,CTLCHR={YES|ASA}]
                         [,DEVICE=nnn]
                         [,ERROPT={RETRY|IGNORE|name}]
                         [,FUNC=xxxx]
                         [,IOAREA2=name]
                         [,IOREG=(r)]
                         [,MODNAME=name]
                         [,PRINTOV=YES]
                         [,RDONLY=YES]
                         [,RECFORM={FIXUNB|VARUNB|UNDEF}]
                         [,RECSIZE=(r)]
                         [,SEPASMB=YES]
                         [,STLIST=YES]
                         [,TRC=YES]
                         [,UCS={ON|OFF}]
                         [,WORKA=YES]
-----------------------------------------------------------
```

DTFPR is used to define an output file for a printer.

ASOCFLE=filename: This operand is used together with the FUNC operand to define associated files for the 2560, 3525, or 5424/5425. (For a description of associated files see VSE/Advanced Functions Application Programming: Macro User's Guide.) ASOCFLE specifies the filename of an associated read and/or punch file, and enables macro sequence checking by the logic module of each associated file. One filename is required per DTF for associated files.

Figure 20 on page 119 defines the filename specified by the ASOCFLE operand for each of the associated DTFs.

BLKSIZE=n: This operand specifies the length of IOAREA1. The maximum values which may be specified in this operand and the lengths assumed when it is omitted are given for the different devices in Figure 21 on page 120 .

CONTROL=YES: This operand is specified if the CNTRL macro will be issued for the file. If this operand is specified, omit CTLCHR. This operand is not allowed for the 2560 or 5424/5425.

CTLCHR={YES|ASA}: This operand is specified if first-character control is used. ASA specifies the American National Standards Institute, Inc. character set. CTLCHR=YES specifies the S/370 char-

Code in FUNC=Operand	Filename Specification in ASOCFLE=Operand of		
	Read DTFCD	Punch DTFCD	Print DTFPR
FUNC=RPW	Filename of Punch DTFCD	Filename of Print DTFPR	Filename of Read DTFCD
FUNC=PW		Filename of Print DTFPR	Filename of Punch DTFCD
FUNC=RW	Filename of Print DTFPR		Filename of Read DTFCD

Examples:

1. If FUNC=PW is specified,
   a. specify the filename of the print DTFPR in the ASOCFLE operand of the punch DTFCD and
   b. specify the filename of the punch DTFCD in the ASOCFLE operand of the print DTFPR.
2. If FUNC=RPW is specified,
   a. specify the filename of the punch DTFCD in the ASOCFLE operand of the read DTFCD, and
   b. specify the filename of the print DTFPR in the ASOCFLE operand of the punch DTFCD, and
   c. specify the filename of the read DTFCD in the ASOCFLE operand of the print DTFPR.

Figure 20. ASOCFLE Operand Usage with Print Associated Files

acter set. See Appendix A for a list of codes. If this parameter is specified, omit CONTROL. This operand must not be specified for the 2560 or 5424/5425.

If CTLCHR=ASA is specified for the 3525, the + character is not allowed. To print on the first line of a card, you must issue either a space 1 command or a skip to channel 1 command. For 3525 print associated files, you must issue a space 1 command to print on the first line of a card.

DEVADDR={SYSLOG|SYSLST|SYSnnn}: This operand specifies the symbolic unit to be associated with the printer. SYSLOG and SYSLST must not be specified for the 2245, 2560, 3525, or 5424/5425.

DEVICE={1403|1443|2245|2560P|2560S|3203|3211|3525|3800|5203|5425P|5425S|PRT1}: This operand specifies which device is used for the file. The 'P' and 'S' included with the "2560" and "5425" parameters specify primary or secondary input hoppers. Specify 5425P/S for 5424(P/S). "PRT1" refers to a 3211 or 3211-compatible printer. For a list of PRT1 printers, see VSE/Advanced Functions, System Control Statements, SC33-6095.

Devices	Maximum length (in bytes) which can be specified[1]	Length assumed (in bytes)[2]
1403-1, -4	100[5]	121
1403-6, -7	120[5]	121
1403-2, -3, -5, -8, -9	132	121
1443	144	121
2560	384	64
3203	132	121
PRT1	132[3]	121
3525	64	64
3800/3200	384 (without TRC)[4]	136 (without TRC)[4]
5203	132	96
5424/5425	128	96

[1] RECFORM is FIXUNB or UNDEF and operand CTLCHR is not specified.
[2] The parameter BLKSIZE=n is omitted.
[3] 150 if the feature is available in a 3211.
[4] For a 3800, the maximum length is 385 if TRC=YES is used, and the assumed length is 137.
[5] Maximum print position of the device.

Notes:

- If CTRCHR=YES/ASA is specified, add 1 byte to the maximum length which can be specified.
- If RECFORM=VARUNB is specified add 4 bytes to the maximum value which can be specified.
- For the 2245, if RECFORM=VARUNB and CTLCHR=YES/ASA are specified, the maximum blocksize is 805 bytes.

Figure 21. Maximum and Assumed Lengths for the IOAREA1

If the DEVICE operand is omitted, 1403 is assumed.

ERROPT={RETRY|IGNORE|name}: This operand specifies the action to be taken in the case of an equipment error. The functions of the parameters are described below.

RETRY
Can be specified for a PRT1 printer only. RETRY indicates that if an equipment check with command retry is encountered, the command is retried once. If the retry is unsuccessful, a message is issued and the job is canceled.

IGNORE
Can be specified only for the 3525. IGNORE indicates that the error is to be ignored. The address of the record in error is put in register 1 and made available for processing. Byte 3, bit 3 of the CCB is also set on (see Figure 3 on page 16); you can check this bit and take the appropriate action to recover from the error. IGNORE must not be specified for files with two I/O areas or a work area.

name
Can be specified only for a PRT1 printer. If an equipment check with command retry is encountered, the command is retried once. If the retry is unsuccessful a message is issued and the job canceled. With other types of errors (for these see the CCB, Figure 3 on page 16) an error message is issued, error information is placed in the CCB, and control is given to your error routine, where you may perform whatever actions are desired. If any IOCS macros are issued in the routine, register 14 must be saved; if the operand RDONLY=YES is specified, register 13 must also be saved. To continue processing at the end of the routine, return to IOCS by branching to the address in register 14.

FUNC={W[T]|RW[T]|RPW[T]|PW|[T]}: This operand specifies the type of file to be processed by the 2560, 3525, or 5424/5425. W indicates print, R indicates read, P indicates punch, and T (for the 3525 only) indicates an optional 2-line printer.

RW[T], RPW[T], and PW[T] are used, together with the ASOCFLE operand, to specify associated files; when one of these parameters, other than T, is specified for a printer file it must also be specified for the associated file(s). Note: Do not use T for associated files; it is valid only for printer files.

If a 2-line printer is not specified for the 3525, multi-line print is assumed. T is ignored if CONTROL or CTLCHR is specified.

IOAREA1=name: This operand specifies the name of the output area.

IOAREA2=name: This operand specifies the name of a second output area.

IOREG=(r): If two output areas and no work areas are used, this operand specifies the register into which IOCS will place the address of the area where you can build a record. For (r) specify one of the registers 2 to 12.

MODNAME=name: This operand may be used to specify the name of the logic module that is used with the DTF table to process the file. If the logic module is assembled with the program, MODNAME must specify the same name as the PRMOD macro. If this operand is omitted, standard names are generated for calling the logic module. If two DTF macros call for different functions that can be handled by a single module, only one module is called.

PRINTOV=YES: This operand is specified if the PRTOV macro is included in your program. This operand is not allowed for the 2560 or 5424/5425.

RDONLY=YES: This operand is specified if the DTF is used with a read-only module. Each time a read-only module is entered, register 13 must contain the address of a 72-byte doubleword-aligned save area. Each task requires its own uniquely defined save area. Each time an imperative macro (except OPEN or OPENR) is issued, register 13 must contain the address of the save area associated with the task. The fact that the save areas are unique for each task makes the module reentrant (that is, capable of being used concurrently by several tasks).

If an ERROPT routine issues I/O macros which use the same read-only module that caused control to pass to either error routine, your program must provide another save area. One save area is used for the normal I/O, and the second for I/O operations in the ERROPT routine. Before returning to the module that entered the ERROPT routine, register 13 must be set to the save area address originally specified for the task.

If this operand is omitted, the module generated is not reenterable and no save area need be established.

RECFORM={FIXUNB|UNDEF|VARUNB}: The operand RECFORM=FIXUNB is specified whenever the record format is fixed. When the record format is FIXUNB, this entry may be omitted.

The entry RECFORM=UNDEF is specified whenever the record format is undefined. If the output is variable and unblocked, enter VARUNB.

RECSIZE=(r): This operand specifies the general register (any one of 2 to 12) that will contain the length of an output record of undefined format. The length of each record must be loaded into the register before issuing the PUT macro.

SEPASMB=YES: Include this operand only if the DTFPR is to be assembled separately. This causes a CATALR card with the filename to be punched ahead of the object deck and the filename to be defined as an ENTRY point in the assembly. If the operand is omitted, the assembler assumes that the DTF is assembled together with the problem program.

STLIST=YES: Include this operand if the selective tape listing feature (1403 only) is used. If this entry is specified, the CONTROL, CTLCHR, and PRINTOV entries are not valid and will be ignored if specified. If this operand is specified, RECFORM must have the parameter FIXUNB.

TRC=YES: This operand applies to the 3800 Printing Subsystem; DEVICE=3800 should be specified. TRC=YES specifies that a table reference character is included as the first byte of each output data line (following the optional print control character). The printer uses the table reference character to select the character arrangement table corresponding to the order in which the table names have been specified with the CHAR parameter on the SETPRT job control statement (or SETPRT macro instruction).

If a printer other than a 3800 is specified on the DEVICE parameter, any table reference character sent to that printer is treated as data.

UCS={OFF|ON}: For a printer with the universal character set feature, or for a 3800 Printing Subsystem, this operand determines whether data checks occurring in case of unprintable characters are indicated to the operator or printed as blanks. The entry is especially useful if you are using first-character forms control and have modules that cannot process the CNTRL macro. If the operand is omitted, OFF is the default.

ON
Data checks are processed with an operator indication.

OFF
Data checks are ignored and blanks are printed for the unprintable character.

WORKA=YES: If output records are processed in work areas instead of in the I/O areas, specify this operand. You must set up the work area in storage. The address of the work area, or a general-purpose register which contains the address, must be specified in each PUT macro.

DTFPT

```
-------------------------------------------------------------
Name         Operation     Operand
-------------------------------------------------------------
[name]       DTFPT         BLKSIZE=n
                           ,DEVADDR=SYSxxx
                           ,IOAREA1=name
                           [,DELCHAR=X'nn']
                           [,DEVICE=nnnn]
                           [,EOFADDR=name]
                           [,EORCHAR=X'nn']
                           [,ERROPT={IGNORE|SKIP|name}]
                           [,FSCAN=name]
                           [,FTRANS=name]
                           [,IOAREA2=name]
                           [,IOREG=(r)]
                           [,LSCAN=name]
                           [,LTRANS=name]
                           [,MODNAME=name]
                           [,OVBLKSZ=n]
                           [,RECFORM=xxxxxx]
                           [,RECSIZE=(r)]
                           [,SCAN=name]
                           [,SEPASMB=YES]
                           [,TRANS=name]
                           [,WLRERR=name]
-------------------------------------------------------------
```

The DTFPT macro is used to define an input or output file on a paper tape I/O device.

BLKSIZE=n: This operand specifies the length of the input or output area. The maximum block size is 32,767 bytes.

DELCHAR=X'nn' This operand specifies the configuration of the delete character and must be used for output files only, that is, when DEVICE=1018 is specified. The constant X'nn' consists of two hexadecimal digits. The delete character is used in the error recovery procedure, and you must specify the correct configuration in accordance with the number of tracks of the output tape, as follows:

X'1F' for five tracks.
X'3F' for six tracks.
X'7F' for seven tracks.
X'FF' for eight tracks.

Note: The delete character is required only if the 1018 has the error correction feature.

DEVADDR=SYSxxx: This operand specifies the logical unit (SYSnnn) associated with this file. An actual channel and unit are assigned to the unit by an ASSGN job control statement. The ASSGN statement contains the same symbolic name as DEVADDR.

DEVICE={2671|1017|1018}: This operand is required only to specify the paper tape I/O device. If this entry is omitted, 2671 is assumed.

EOFADDR=name: This operand specifies the name of your end-of-file routine. IOCS automatically branches to this routine on an end-of-file condition if the end-of-file switch is set on. The routine can execute any operation required for the end-of-file, issue the CLOSE macro for the file, or return to IOCS by branching to the address in register 14. In the latter case, IOCS reads in the next record. The end-of-file condition cannot occur on the 1018.

EORCHAR=X'nn': This operand specifies the user-defined end-of-record (EOR) character, where nn is two hexadecimal digits. It must be used for output files with undefined record format only. IOCS writes this character after the last character of the undefined record.

ERROPT={IGNORE|SKIP|name}: This operand is specified if you do not want a job terminated when the standard recovery procedure cannot recover from a read or write error. If the ERROPT entry is omitted and a read or write error occurs, IOCS terminates the job.

For input files, IGNORE allows IOCS to handle the record as if no errors were detected. If SKIP is specified, IOCS skips the record in error and reads the next record.

For output files with shifted codes, ERROPT cannot be specified. For unshifted codes, the options ERROPT=IGNORE and ERROPT=name can be specified. IGNORE allows IOCS to handle the record as if no errors were detected.

The ERROPT=SKIP option is ignored and causes IOCS to terminate the job.

If two I/O areas are used, the CLOSE macro checks the last record, and the option ERROPT=name is treated as option ERROPT=IGNORE.

For name, specify the symbolic address of your error routine that will process errors. On an error condition, IOCS reads or writes the complete record, including the error character(s), and then branches to the error routine. At the end of the error routine, return to IOCS by branching to the address in register 14. The next record is then read or written. You must not issue any GET or PUT macros for records in the error block. If the error routine contains any other IOCS macros, the contents of register 14 must be saved and restored.

FSCAN=name: This operand must be included for every output file using a shifted code. Omit this operand for an input file. The operand specifies the name of a scan table in your program used to select groups of figures. This table must conform to the specifications of the machine instruction TRT. The entry in the table for each letter character must be the letter shift character, and all other entries must be hexadecimal zero. Any deviation from this results in incorrect translation.

FTRANS=name: This operand must be included for every input file using a shifted code and is not permitted for output files. It specifies the name of a figure shift table in your program. This table must conform to the specifications of the machine instruction TR.

IOAREA1=name: This operand specifies the name of an input or output area.

IOAREA2=name: This operand specifies the name of a second input or output area. When this operand is specified, IOCS overlaps the I/O operation in one area with the processing of the record in the other.

IOREG=(r): This operand must be included if two input or output areas are used. For input, it specifies the register into which IOCS puts the address of the logical record available for processing. For output, it specifies the register that contains the address of the area in which your program can build a record. Any register from 2 to 12 may be specified.

LSCAN=name: This operand must be included for every output file using a shifted code and is not permitted for input files. It specifies the name of a scan table in your program used to select groups of letters. This table must conform to the specifications of the machine instruction TRT. The entry in the table for each figure character must be the figure shift character, and all other entries must be hexadecimal zero. Any deviation from this results in incorrect translation.

LTRANS=name: This operand must be included for every input file using a shifted code and is not permitted for output files. It specifies the name of a letter shift table in your program. This table must conform to the specifications of the machine instruction TR.

MODNAME=name: This operand specifies the name of the logic module used with the DTF table to process the file. If the logic module is assembled with the program, the MODNAME operand in this DTF must specify the same name as the PTMOD macro. If the operand is omitted, IOCS generates standard names for calling the logic module.

OVBLKSZ=n: For input files, this operand specifies the number of characters to be read (before translation and compression) to produce the number of characters specified in the BLKSIZE entry. OVBLKSZ is used only when SCAN=name and RECFORM=FIXUNB are both specified. If OVBLKSZ is omitted, IOCS assumes that the number of characters to be read is equal to the number specified in the BLKSIZE entry. The maximum value is 32,767 bytes.

For output files, OVBLKSZ specifies the number of characters indicated in the BLKSIZE entry, plus the number of shift characters to be inserted. If the size of OVBLKSZ is large enough to allow the insertion of all the shift characters required to build the output record, a single WRITE operation results from a PUT macro. On the other hand, if the size of OVBLKSZ (which must be at least one position larger than BLKSIZE) does not permit the insertion of all the shift characters, several WRITE operations result from a PUT macro. OVBLKSZ is used only when LSCAN and FSCAN are specified with the FIXUNB format. If OVBLKSZ is specified with UNDEF format, it is ignored.

RECFORM={FIXUNB|UNDEF}: This operand specifies the record format for the file. Specify either format for shifted or unshifted codes. If the record format is FIXUNB, this entry may be omitted.

RECSIZE=(r): This operand specifies the number of a register (any one of 2 to 12) that contains the length of the input or output record. This entry is optional for input files. If present, IOCS loads the length of each record read into the specified register. If input files contain shift codes or other characters requiring deletion, IOCS loads the compressed record length into the specified register.

For output files, this entry must be included for undefined records. Before translation, your program must load each record length into the designated register before issuing the PUT macro for the record.

SCAN=name: This operand must be included for all input files using shifted codes. It may also be included if you wish to delete certain characters from each record. The SCAN entry specifies the symbolic name of a table provided by your program. This table must conform to the specifications of the machine instruction TRT. It must contain nonzero entries for all delete characters and, where appropriate, for the figure and letter shift characters.

The table entry for the figure shift character must be X'04'; for the letter shift character, the entry must be X'08'; delete entries must be X'0C'. All other entries in the table must be X'00'. Otherwise, incorrect translation results and a program check may occur.

The table must be large enough to hold the maximum value of coding for the tape being processed; that is, 255 bytes for 8-track tape. This prohibits erroneous coding on the tape from causing a scan function beyond the limits of the scan table.

SEPASMB=YES: Include this operand only if the DTFPT is assembled separately. This causes a CATALR card with the filename to be punched ahead of the object deck and the filename to be defined as an ENTRY point in the assembly. If the operand is omitted, the assembler assumes that the DTF is being assembled with the problem program and no CATALR card is punched.

TRANS=name: The TRANS operand specifies the symbolic name of a table provided within your program. This table must conform to the specifications of the machine instruction TR. For input files, include this entry if a nonshifted code is to be translated into internal system code. Omit the FTRANS and LTRANS entries if this entry is present. If none of these three entries is present, no translation takes place. For output files, include this entry if the internal system code is translated into a shifted or nonshifted code, depending on whether the FSCAN and LSCAN entries are present or omitted.

WLRERR=name: Applies only to paper tape input files when RECFORM=UNDEF is specified.

When IOCS finds a wrong-length record, it branches to the symbolic name specified in the WLRERR entry. If this entry is omitted and the ERROPT entry is included, IOCS considers the error uncorrectable and uses the ERROPT option specified. Absence of both ERROPT and WLRERR entries causes the wrong-length record to be accepted as a normal record. IOCS detects overlength undefined records when the incoming record fills the input area. The input area must, therefore, be at least one position longer than the longest record anticipated.

At the end of the WLRERR routine, return to IOCS by branching to the address in legister 14. The next IOCS read operation will normally cause the remainder of the overlength, undefined record to be read. If any other IOCS macros are included in the record-length error routine, the contents of register 14 must be saved and restored.

> **Note:** A wrong-length condition appears during the first read operation on a 1017 if the combined length of the tape leader and the first record is greater than the length of the longest record anticipated (the length specified in BLKSIZE).

DTFSD

```
-------------------------------------------------------------
Name          Operation     Operand
-------------------------------------------------------------
[name]        DTFSD         BLKSIZE=n
                            ,EOFADDR=name
                            [,CISIZE=n]
                            [,DELETFL=NO]
                            [,DEVADDR=SYSxxx]
                            [,ERROPT={IGNORE|SKIP|name}]
                            [,FEOVD=YES]
                            [,HOLD=YES]
                            [,IOAREA1=name]
                            [,IOAREA2=name]
                            [,IOREG=(r)]
                            [,LABADDR=name]
                            [,PWRITE=YES]
                            [,RECFORM=xxxxxx]
                            [,RECSIZE={n|(r)}]
                            [,SEPASMB=YES]
                            [,TRUNCS=YES]
                            [,TYPEFLE={INPUT|OUPUT|WORK}]
                            [,UPDATE=YES]
                            [,VARBLD=(r)]
                            [,VERIFY=YES]
                            [,WLRERR=name]
                            [,WORKA=YES]
-------------------------------------------------------------
```

The DTFSD macro defines a DASD file for sequential (consecutive) processing. Only IBM standard label formats are processed.

BLKSIZE=n: Enter the length of the I/O area. If the record format is variable or undefined, enter the length of the I/O area needed for the largest block of records.

For input files with fixed-length blocked records, BLKSIZE must be an integer multiple of RECSIZE; for output files, eight bytes must be added for IOCS to construct a count field.

If the file is on an FBA device, the DTF BLKSIZE determines the logical block size. For FBA DASD, the maximum value is 32,761 (that is, seven bytes less than the maximum CISIZE). The BLKSIZE value for output files must include eight bytes for a count field to provide compatibility between FBA and CKD DASD.

The DTFSD BLKSIZE specification can be overridden by the BLKSIZE operand of the DLBL job control statement if RECFORM=xxxBLK. For an output file, the records are blocked according to the size specified by the appropriate BLKSIZE operand (from the DLBL statement if it was specified; otherwise from the DTFSD). For an input file, the BLKSIZE specification must match the format of the data as it resides on the disk.

To use the DLBL BLKSIZE operand:

- The device must be a CKD device or it is ignored.
- Partition GETVIS space for a DTF extension and new buffers must be available.
- DTFSD RECFORM=xxxBLK must have been specified.

CISIZE=n: This operand specifies the control interval size for an FBA device assigned to a non-system file logical unit. If assigned to a system file (SYSRDR, SYSIPT, SYSLST, or SYSPCH), or to a CKD DASD file, the operand is ignored. The value n must be a multiple of the FBA block size and, if greater than 8K, must be a multiple of 2K. The maximum value is 32,768 (32K) except when assigned to SYSLST or SYSPCH, when the maximum is 30,720 (30K).

If CSIZE is omitted, CISIZE=0 is assumed. For an FBA device, control interval size may be overridden for an output file at execution time by specifying the CISIZE parameter on the DLBL control statement. For an input file, the CISIZE value in the format-1 label is used. If zero, then OPEN calculates a value based on the BLKSIZE specification on the DTF.

DELETFL=NO: Specify this operand if the CLOSE macro is not to delete the format-1 and format-3 label for a work file. The operand applies to work files only.

DEVADDR=SYSxxx: This operand must specify the symbolic unit associated with the file if an extent is not provided. A job control EXTENT statement is not required for single-volume input files. If an EXTENT statement is provided, its specification overrides any DEVADDR specification. SYSnnn represents an actual I/O address, and is used in the ASSGN job control statement to assign the actual I/O device address to this file.

EOFADDR=name. This operand specifies the name of your end-of-file routine (for input or work files). IOCS automatically branches to this routine on an end-of-file condition. In this routine, you can perform any operations required at end of file (you generally issue the CLOSE macro).

ERROPT={IGNORE|SKIP|name}: This operand is specified if a job is not to be terminated when a read or write error cannot be corrected in the disk error routines. The disk error routines normally retry failing I/O operations several times before considering the error unrecoverable. Once the error is considered unrecoverable, the job is terminated unless the ERROPT operand is specified. The functions of the parameters are explained below.

IGNORE

The error condition is ignored. The records are made available for processing. When reading spanned records, the whole spanned record or block of spanned records is returned, rather than just the one physical record in which the error occurred.

On output, the physical record or control interval in which the error occurred is ignored as if it were written correctly. If possible, any remaining spanned record segments are written.

SKIP

No records in the error block or control interval are made available for processing. The next block or control interval is read from the disk, and processing continues with the first record of that block. When reading spanned records, the whole spanned record or block of spanned records is skipped, rather than just one physical record.

On an UPDATE=YES file, the physical record or control interval in which the error occurred is ignored as if it were written correctly. If possible, any remaining spanned record segments are written.

name

IOCS branches to your error routine named by this parameter. In this routine you can process or make note of the error condition as desired, but you should not issue any imperative macro instructions for the file invoking the error exit.

FEOVD=YES: This operand is specified if a forced end of volume for disk feature is desired. It forces the end-of-volume condition before physical end of volume occurs. When the FEOVD macro is issued, the current volume is closed, and I/O processing continues on the next volume. This operand does not apply to work files.

HOLD=YES: This operand may be specified only if the track hold function was specified at system generation time and if it is employed when a data input file or a work file is referenced for updating.

IOAREA1=name: This operand specifies, for an input or output file, the symbolic name of the I/O area used by the file. It is not required if WORKA=YES or IOREG=(r) is specified for any input or output file.

If both IOAREA1=name and WORKA=YES are specified on an FBA file, IOAREA1 is ignored.

> **Note:** If the BLKSIZE is overridden by the DLBL statement, and the value is greater than the value specified in the DTF, OPEN issues a GETVIS for the space of the larger I/O area and the specified one is not used.

For variable-length or undefined records, this area must be large enough to contain the largest block or record.

IOAREA2=name: If two I/O areas are used by GET or PUT, this operand is specified. When variable length records are processed, the size of the I/O area must include four bytes for the block size. For output files, the I/O area must include eight bytes. This operand is ignored if IOAREA1 is not specified.

IOREG=(r): This operand specifies, for an input or output file, the general purpose register (any of 2 to 12) in which IOCS puts the address of the logical record that is available for processing. At OPEN time, for output files, IOCS puts into the register specified the address of the area where you can build a record. The same register may be used for two or more files in the same program, if desired. If this is done, the program must store the address supplied by IOCS for each record.

This operand must be specified if

- Blocked input or output records are processed in one I/O area, or
- If two I/O areas are used and the records are processed in both I/O areas.

For an FBA file, the register specified by IOREG will point directly to data in the control interval buffer.

LABADDR=name: Specifies, for an input or output file, the name of the routine in which you process your own labels.

PWRITE=YES: This operand is specified if formatting output operations to an FBA device (PUT for data files or WRITE SQ for work files) are to cause a physical write for each logical block. If omitted, the physical write takes place only when the control interval buffer is full.

RECFORM={FIXUNB|FIXBLK|VARUNB|VARBLK|SPNUNB|SPNBLK|UNDEF}: This operand specifies the type of records for input or output. Enter one of the following parameters:

FIXUNB	For fixed-length unblocked records
FIXBLK	For fixed-length blocked records
VARUNB	For variable-length unblocked records
VARBLK	For variable-length blocked records
SPNUNB	For spanned variable-length unblocked records
SPNBLK	For spanned variable-length blocked records
UNDEF	For undefined records

If RECFORM=SPNUNB or RECFORM=SPNBLK is specified and RECSIZE=(r) is not specified, an assembler diagnostic (MNOTE) is issued, and register 2 is assumed. If WORKA=YES is omitted, an MNOTE is issued and WORKA=YES is assumed. If RECFORM is omitted, FIXUNB is assumed.

If RECFORM=xxxBLK is specified, you can override the BLKSIZE value with the BLKSIZE operand on the DLBL statement at execution time.

For work files, use FIXUNB or UNDEF only.

RECSIZE={n|(r)}: For fixed-length blocked records, RECSIZE is required. It specifies the number of characters in each record.

Register notation must be used when processing spanned or undefined records. When processing undefined records and variable-length spanned records, RECSIZE is required for output files and is optional for input files. The operand is invalid for work files. It specifies a general register (any one of 2 to 12) that contains the length of the record. On output, you must load the length of each record into the designated register before issuing a PUT macro. If specified for input, IOCS provides the length of the record transferred to virtual storage.

SEPASMB=YES: Include this operand only if the DTFSD is to be assembled separately. This causes a CATALR card with the filename to be punched ahead of the object deck and the filename to be defined as an ENTRY point in the assembly. If the operand is omitted, the assembler assumes that the DTF is assembled together with the problem program. The operand does not apply to work files.

TRUNCS=YES: This operand is specified if FIXBLK DASD files contain short blocks embedded within an input file or if the input file was created with a module that specified TRUNCS. This entry is also specified if the TRUNC macro is issued for a FIXBLK output file. The operand does not apply to work files.

TYPEFLE={INPUT|OUTPUT|WORK}: Use this operand to indicate whether the file is an input or an output or a work file. If INPUT or OUTPUT is specified, the GET or PUT macros, respectively, can be used. If WORK is specified, the READ and WRITE, NOTE and POINTx, and CHECK macros must be used, and RECFORM must be either FIXUNB or UNDEF. If the operand is omitted, INPUT is assumed.

UPDATE=YES: This operand must be included if the DASD input or work file is updated - that is, if disk records are read, processed, and then re-written in the same disk record locations from which they were read. CLOSE writes any remaining records in sequence onto the disk.

This operand is invalid for a file on a DASD assigned to a system logical unit (SYSRDR, SYSIPT, SYSLST, or SYSPCH). If a PUT is attempted to an input file, the job will be terminated.

VARBLD=(r): Whenever variable-length blocked records are built directly in the output area (no work area specified), this entry must be included. It specifies the number (r) of a general-purpose register (any one of 2 to 12), which will always contain the length of the available space remaining in the output area.

IOCS calculates the space still available in the output area, and supplies it to you in the designated register after the PUT macro is issued for a variable-length record. You then compare the length of your 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, issue a TRUNC macro to transfer the completed block of records to the file. Then, the present record is built at the beginning of the output area in the next block.

VERIFY=YES: This operand is included if you want to check the parity of disk records after they are written. If this operand is omitted, any records written on a disk are not verified.

WLRERR=name: This operand applies only to disk input files. It does not apply to undefined records. WLRERR specifies the symbolic name of your routine to receive control if a wrong-length record is read.

If the WLRERR operand is omitted but a wrong-length record is detected by IOCS, one of the following conditions results:

- If the ERROPT entry is included for this file, the wrong-length record is treated as an error block and handled according to your specifications for an error (IGNORE, SKIP, or name of error routine).
- If the ERROPT entry is not included, the error is ignored.

Undefined records are not checked for incorrect record length. The record is truncated when the BLKSIZE specification is exceeded.

WORKA=YES: If records of an input or output file are processed or built in work areas instead of I/O areas, specify this operand. You must set up the work area in storage. The address of the work area, or a general-purpose register which contains the address, must be specified in each GET or PUT macro. For a GET or PUT macro, IOCS moves the record to, or from, the specified work area. WORKA=YES is required for SPNUNB and SPNBLK. When this operand is specified for a file, the IOREG operand must be omitted. For spanned records, the work area must be sufficiently long to hold the longest spanned record.

DTL

```
------------------------------------------------------------
Name          Operation     Operand
------------------------------------------------------------
[name]        DTL           NAME=resourcename
                            [,CONTROL={E|S}]
                            [,LOCKOPT={1|2}]
                            [,KEEP={NO|YES}]
                            [,OWNER={TASK|PARTITION}]
                            [,SCOPE={INT|EXT}]
------------------------------------------------------------
```

The DTL (Define The Lock) macro generates a control block which is used by the LOCK/UNLOCK macros to enqueue/dequeue a resource access request. The control block, commonly called 'DTL', is generated at the time of program assembly.

NAME=resourcename: Specifies the name by which the resource is known to the system for the purpose of access share control. It is by this name that the system controls shared access of the resource as requested by active tasks via the LOCK macro. These tasks may all be active in one partition, or they may be distributed over several partitions; the resource-share control extends across partitions.

The name may be up to twelve bytes long. If it is shorter, it is padded with blanks. Note that the name must not begin with any of the characters A through I or V, because these characters are reserved for IBM usage.

CONTROL={E|S}: Defines how the named resource can be shared while your program owns it, which is determined by this specification and your specification for the operand LOCKOPT. A specification of E means the resource is enqueued for exclusive use; a specification of S means the resource is enqueued as shareable.

LOCKOPT={1|2}: This operand, together with the CONTROL parameter, determines how the system controls shared access in response to a LOCK request.

- LOCKOPT=1 and CONTROL=E: No other task is allowed to use the resource concurrently.
- LOCKOPT=1 and CONTROL=S: Other 'S' users are allowed concurrent access, but no concurrent 'E' user is allowed.
- LOCKOPT=2 and CONTROL=E: No other 'E' user gets concurrent access; however, other 'S' users can have access to the resource.
- LOCKOPT=2 and CONTROL=S: Other 'S' users can have concurrent access and, in addition, one 'E' user is allowed.

All users of a particular resource have to use the same LOCKOPT specification when they lock the resource. (Exception: if LOCKOPT=1 and CONTROL=E, the lock status may be modified.)

KEEP={NO|YES}: This operand may be used to lock the named resource beyond job step boundaries. Only a main task should use this operand. KEEP=NO indicates that the named resource once locked, is to be released automatically at the end of the particular job step. With KEEP=YES, a named resource that is locked remains locked across job steps; it will be automatically released at end-of-job.

If a job terminates abnormally, all resources with KEEP=YES are unlocked by the abnormal termination routine.

OWNER={TASK|PARTITION}: Defines whether the named resource, once locked, can be unlocked only by the task which issued the corresponding LOCK request (OWNER=TASK), or whether it can be unlocked by any task within the partition (OWNER=PARTITION).

When OWNER is defined as PARTITION, a LOCK request for the resource must not specify FAIL=WAIT or FAIL=WAITC because deadlock prevention (return code 16) is not supported with OWNER=PARTITION.

SCOPE={INT|EXT}: This operand may be used for locking resources across systems. SCOPE=EXT specifies that the lock is used across systems. You may omit the parameter if you want to lock your resources only on one system since the default is SCOPE=INT (that is, the locking applies to one system only).

DUMODFx

Name	Operation	Operand
[name]	DUMODFx	ERREXT=YES ,ERROPT=YES [,RDONLY=YES] [,SEPASMB=YES]

The DUMODFx macro defines a logic module for a diskette file.

Two categories of file characteristics are defined for diskette unit module generation macros:

- DUMODFI - Diskette Unit MODule, Fixed length records, Input file.

- DUMODFO - Diskette Unit MODule, Fixed length records, Output file.

ERREXT=YES: Include this operand if permanent errors are returned to a problem program ERROPT routine or if the ERET macro is used with the DTF and module. The ERROPT operand must be specified for this module.

ERROPT=YES: This operand applies to both DUMODFx macros. This operand is included if the module handles any of the error options for an error chain. Logic is generated to handle any of the three options (IGNORE, SKIP, or name) regardless of which option is specified in the DTF. This module also processes any DTF in which the ERROPT operand is not specified.

If this operand is not included, your program is canceled whenever a permanent error is encountered.

RDONLY=YES: This operand causes a read-only module to be generated. If this operand is specified, any DTF used with this module must have the same operand.

SEPASMB=YES: Include this operand only if the logic module is to be assembled separately. This causes a CATALR card with the filename to be punched ahead of the object deck and the filename to be defined as an ENTRY point in the assembly. If the operand is omitted, the assembler assumes that the DTF is assembled together with the problem program.

DUMODFx

Standard DUMOD Names

Each name begins with a 3-character prefix (IJN) and continues with a 5-character field corresponding to the options permitted in the generation of the module, as shown below.

DUMODFx name = IJNabcde

a	= D	always
b	= I	DUMODFI
	= O	DUMODFO
c	= C	ERROPT=YES and ERREXT=YES
	= E	ERROPT=YES
	= Z	neither is specified
d	= Z	always
e	= Y	RDONLY=YES
	= Z	RDONLY not specified

Subset/Superset DUMOD Names

The following chart shows the subsetting and supersetting allowed for DUMOD names.

```
-----------------------------------------
            * + * *
    I J N D I C Z Y
            O E   Z
              Z
-----------------------------------------
  + Subsetting/supersetting permitted

  * No subsetting/supersetting permitted
-----------------------------------------
```

DUMP

```
--------------------------------------------------
Name        Operation    Operand
--------------------------------------------------
[name]      DUMP
--------------------------------------------------
```

This macro provides a hexadecimal dump of the following:

- The contents of the entire supervisor area and the used part of the system GETVIS area, or of some supervisor control blocks only (see Note below).
- The contents of the partition that issued the macro.
- The contents of the registers.

Note: The dump includes the contents of some supervisor control blocks only, rather than the entire supervisor area, if the STDOPT job control command specifies DUMP=PART or NO, or if a job control statement // OPTION PARTDUMP or NODUMP is submitted.

In addition, the macro causes the job step to be terminated if DUMP was issued by the main (or only) task of the program. If DUMP was issued by a subtask, the macro causes that subtask to be detached without terminating the main task in the partition.

The dump provided by the macro is always directed to SYSLST, which must be opened if disk or tape; if SYSLST is a tape, that tape must be positioned as desired.

If DUMP is issued by a job running in real mode, the storage contents of the partition are dumped only up to the limit as determined by the SIZE parameter of the EXEC job control statement, plus the storage obtained dynamically through the GETVIS macro. If SIZE was not specified, the entire partition will be dumped. If DUMP is issued by a program running in virtual mode, the entire partition is dumped.

ENDFL

Name	Operation	Operand
[name]	ENDFL	{filename|(0)}

The ENDFL (end file load mode) macro ends the mode initiated by the SETFL macro. The ENDFL macro must be issued only after a SETFL and before a CLOSE.

The ENDFL macro performs an operation similar to CLOSE for a blocked file. It writes the last block of data records, if necessary, and then writes an end-of-file record after the last data record. Also, it writes any index entries that are needed followed by dummy index entries for the unused portion of the prime data extent.

filename|(0): The name of the file to be loaded is the only parameter required, and is the same as the name specified in the DTFIS header entry for the file. The filename can be specified either as a symbol or in register notation. Register notation is necessary if your program is to be self-relocating.

ENQ

```
-----------------------------------------------------
Name        Operation    Operand
-----------------------------------------------------
[name]      ENQ          {rcbname|(0)}
-----------------------------------------------------
```

A task protects a resource by issuing an ENQ (enqueue) macro. When the RCB, (identified by the rcbname) is enqueued, the task requesting the resource is either queued and executed, or if the requested resource is held by another task, is placed in a wait condition. When the task holding that resource completes, that task issues the DEQ (dequeue) macro. All other tasks that are then waiting for the dequeued resource are freed from their wait condition, and the highest priority task either obtains or maintains control.

If a task is terminated without dequeuing its queued resources, any task subsequently trying to enqueue that resource is abnormally terminated. If a task issues two ENQs without an intervening DEQ for the same resource, the task is canceled. Also, any task that does not control a resource but attempts to dequeue that resource is terminated, unless DEQ appears in the abnormal termination routine. If DEQ appears in the abnormal termination routine, it is ignored.

Although the main task does not require the program to set up an intertask communication ECB to enqueue and dequeue, every subtask using that facility must have the ECB operand in the ATTACH macro, and that ECB must not be used for any other purpose. Also, a resource can be protected only within the partition containing the ECB.

Note: Do not use the ENQ macro in your AB exit routine for a resource that is held by the main task, since a deadlock may occur.

Name	Operation	Operand
[name]	EOJ	

Issue the EOJ macro in the main task or in the only program within a partition, to inform the system that the job step is finished. If a subtask issues an EOJ, the subtask is detached and the remainder of the partition continues. If the main task issues EOJ, all abnormal termination exits (via STXIT AB) are taken for the subtasks still attached.

ERET

Name	Operation	Operand
[name]	ERET	{SKIP|IGNORE|RETRY}

This macro enables your program's ERROPT or WLRERR routine to return to IOCS and specify an action to be taken. The macro applies to DTFSD files and to DTFIS, DTFMT and DTFDU files with the ERREXT operand specified.

SKIP: Passes control back to the logic module to skip the block of records or control interval in error and process the next one. For disk or diskette output, an ERET SKIP is treated as an ERET IGNORE.

IGNORE: Passes control back to the module to ignore the error and continue processing.

RETRY: Causes the module to retry the operation that resulted in the error. With MTMOD for any error or with SD wrong-length record errors, RETRY cancels the job with an invalid SVC message.

ESETL

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

The ESETL (end set limit) macro ends the sequential mode initiated by the SETL macro. If the records are blocked, ESETL writes the last block back if a PUT was issued.

filename|(1): Is the same name as the name specified in the DTFIS header entry. The name can be specified as a symbol or in register notation. Register notation is necessary if your program is to be self-relocating.

> **Note:** If ADDRTR and/or RANSEQ are specified in the same DTF, ESETL should be issued before issuing a READ or WRITE; another SETL can be issued to restart sequential retrieval. Sequential processing must always be terminated by issuing an ESETL macro.

EXCP

```
------------------------------------------------
Name        Operation    Operand
------------------------------------------------
[name]      EXCP         {blockname|(1)}[,REAL]
------------------------------------------------
```

The EXCP (execute channel program) macro requests physical IOCS to start an input/output operation for a particular I/O device.

Physical IOCS determines the device from the CCB or IORB control block specified by blockname. Physical IOCS places the block in a queue of such blocks and returns control to the problem program. Physical IOCS causes the channel program to be executed as soon as the channel and device are available. I/O interruptions are used to process I/O completion and to start I/O requests if the channel or device was busy at the time the EXCP was executed.

blockname|(1): Is the virtual address of the control block established for the device. It can be given as a symbol or in register notation.

REAL: Indicates that the addresses in the CCWs and the address in the control block pointing to the first CCW have already been translated into real addresses; the operand causes the CCW translation routine to be skipped. (For a program running in real mode, the operand is ignored.)

In your program, the EXCP macro with the REAL operand must be preceded by the PFIX macro that causes the system

- to page in those program pages which contain the pertinent control block, channel program, I/O areas, and IDA (indirect address) words (if used) and
- to fix these pages in their page frames.

Notes:

1. In S/370 mode with option REAL, if the I/O area being used crosses page boundaries, the data address in the appropriate CCW(s) must point to the required indirect data address words within your program; in addition, bit 37 (the IDA bit) of these CCWs must be set to 1. If REAL is not specified, the IDA bit must be set to 0.

2. A channel program has to start with:

   - a long seek command in the case of a CKD DASD.
   - a define extent command in the case of an FBA DASD.

   The data chaining and, in S/370 mode, also the IDA bit must be set to zero for these commands.

EXIT

Name	Operation	Operand
[name]	EXIT	{AB|IT|MR|OC|PC|TT}

The EXIT macro is used to return control from your exit control or MR routine to the instruction in your interrupted program immediately after the instruction where the interruption occurred. For AB, control is returned to the instruction following the EXIT AB macro. Your routine is specified in the STXIT macro (for MR, in the DTFMR macro). The operands have the following meanings:

AB	Exit from your abnormal task termination routine of your main task.
IT	Exit from your interval timer routine.
MR	Exit from your stacker selection routine (MICR document processing) to the supervisor.
OC	Exit from your routine which handles the operator attention interrupt.
PC	Exit from your program check routine.
TT	Exit from your task timer routine.

The EXIT macro should be issued only in the corresponding (AB, IT, MR, OC, PC, TT) routine; a program check may occur if this rule is not observed.

Detailed information on the save area and the interrupt status is given in VSE/Advanced Functions Diagnosis: Service Aids, SC33-6099.

For AB, the cancel condition and ABEND indication of the affected task are reset. The EXIT AB macro may be used only in main tasks. In a subtask, it would result in an illegal SVC. You have to make sure that the abnormal termination condition has been cleared up by your abnormal task termination routine before using the EXIT AB macro.

For IT, OC, PC, and TT, the interrupt status information and registers are restored from the save area; thus, the save area contents are not over-written.

EXTRACT

```
------------------------------------------------------------
Name         Operation    Operand
------------------------------------------------------------
To display partition boundaries:

[name]       EXTRACT      ID=BDY
                          ,AREA={name1|(S,name1)|(r1)}
                          ,LEN={length|(r2)}
                          [,MFG={name3|(r3)}]
                          [,MODE={T|P}]

To display unit information:

[name]       EXTRACT      ID=PUB
                          ,AREA={name1|(S,name1)|(r1)}
                          ,LEN={length|(r2)}
                          [,MFG={name3|(r3)}]
                          [,PID={name4|(S,name4)|(r4)}]
                          [,SEL={name5|(S,name5)|(r5)}]
                          [,DISP={name6|(S,name6)|(r6)}]
------------------------------------------------------------
```

The EXTRACT macro may be used to retrieve and display partition boundaries or unit information (from the PUB table). The information retrieved can be interpreted with the help of the two macros MAPBDY and IJBPUB.

ID={BDY|PUB}: Specifies the information to be retrieved. BDY extracts partition boundaries, whereas PUB displays unit information.

AREA={name1|(S,name1)|(r1)}: Specifies the address of the area where the extracted information is to be stored.

LEN={length|(r2)}: Specifies the length of the area as an integer, a selfdefining term, or as a value in a register. The default length is 1 byte.

MFG={name3|(r3)}: The MFG operand is required if the program is to be reenterable. It specifies the address of a 64-byte dynamic storage area, that is: storage which your program obtained through a GETVIS macro. The area is required for system use during execution of the macro.

PID={name4|(S,name4)|(r4)}: Specifies the address of a two-byte field containing the PIK of the partition for which the information is retrieved. If this operand is omitted, the identifier of the partition issuing the request is taken as the default.

SEL={name5|(S,name5)|(r5)}: Specifies the address of a halfword containing the logical unit number in the same format as the logical unit number in the CCB.

DISP={name6|(S,name6)|(r6)}: Defines the offset within the PUB table entry of the specified device. DISP may be specified as a number, a register containing the displacement value, or the field name in the DSECT generated by the IJBPUB macro.

MODE={T|P}: MODE=T indicates that the temporary boundaries of the issuing partition are to be returned. PID may not be specified in this case, since a snapshot of any other partition's temporary boundaries is unreliable.

If the partition is executing in real mode, the boundaries of the real partition (which, in ECPS:VSE mode, is contained in the corresponding virtual partition) are returned.

MODE=P indicates that the permanent boundaries of the issuing partition or the partition indicated by PID are to be returned. They correspond to the latest allocation and may not have been used by the active job yet.

Return Codes in Register 15

0	The requested information has been extracted.
4	The partition specified is not supported in the system.
8	The logical unit specified exceeds the range of the logical units for the specified partition.
12	The LUB is not assigned or ignored.
16	The length parameter is found to be zero, negative, or below the minimum; or the DISP specification exceeds the length of the PUB entry.

Output from EXTRACT ID=BDY

The output from the EXTRACT macro is described by the following mappings, MAPBDY and MAPBDYVR.

1. Temporary Boundaries (MODE=T):

```
[label] MAPBDY  [DSECT=YES]
```

MAPBDY:

Label	Bytes	Description
PBEGIN	4	Partition start address, corresponding to problem program save area address (field PIB SAVE).
PENDLOG	4	Logical end of partition (last addressable byte, GETVIS area excluded), corresponding to field

		PPEND in the partition communication region.
PGEND	4	Physical end of partition (last addressable byte, GETVIS area included).
PFIXLMT	4	PFIX limit (K-bytes) or zero (real mode).
PFIXCNT	4	PFIX count (number of PFIXed pages).
MBDYLEN		EQU*-PBEGIN MAPBDY area length

2. Permanent Boundaries (MODE=P):

   [label] MAPBDYVR [DSECT=YES]

   MAPBDYVR:

Label	Bytes	Description
VPBEGIN	4	Virtual partition start address (corresponding to latest allocation).
VPEND	4	Virtual partition logical end address (last addressable byte, GETVIS area excluded).
VPGEND	4	Virtual partition physical end address (last addressable byte, GETVIS area included, corresponding to latest allocation).
RPBEGIN	4	Real partition start address (corresponding to latest allocation).
RPEND	4	Real partition end address (last addressable byte, corresponding to latest allocation).
VBDYLEN		EQU*-VPBEGIN MAPBDYVR area length

If 'label' is omitted, the default is MAPBDY and MAPBDYVR.
DSECT=YES generates a DSECT; if omitted, inline code is generated.

FCEPGOUT

You can code the macro in either of the following two formats:

```
--------------------------------------------------
Name        Operation    Operand
--------------------------------------------------
[name]      FCEPGOUT     beginaddr,endaddr
                         [,beginaddr,endaddr]

[name]      FCEPGOUT     {listname|(1)}
--------------------------------------------------
```

The FCEPGOUT macro causes a specific area in real storage to be paged-out at the next page fault. This request is ignored if the specified area does not contain a full page. This can happen up to an area size of 4K minus 2 bytes (see Figure 22).

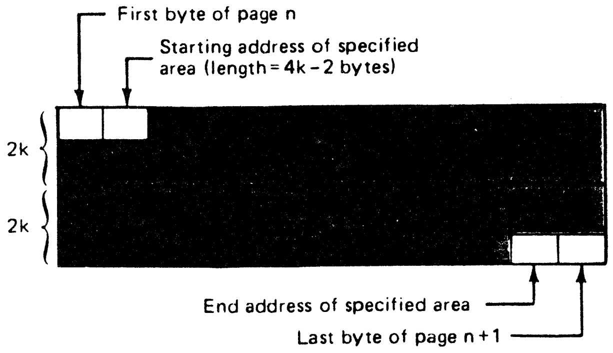

First byte of page n
Starting address of specified area (length=4k-2 bytes)
2k
2k
End address of specified area
Last byte of page n+1

Figure 22. Worst Case of an Area Not Containing one Full Page

If your supervisor was generated with VM=YES (in the SUPVR generation macro), execution of the macro results in a null operation; the return code is set to zero.

beginaddr: Points to the first location of the area to be paged out.

endaddr: Points to the last location of the area to be paged out.

listname|(1): Is the symbolic name of a list of consecutive 8-byte entries as shown below.

```
┌──────────────────────────────────────────────────┐
│ X'00' │ address constant   │  length minus 1     │
└──────────────────────────────────────────────────┘
0       1                    4                     7
```

where:

address constant = Address of the first byte of the area to be paged out.

length = A binary constant indicating the length of the area to be paged out.

A non-zero byte following an entry indicates the end of the list. Register notation may also be used.

Exceptional Conditions

- The program is running in real mode.
- The page(s) referenced by the macro is (are) outside of the requesting partition.
- A page handling request is pending for the referenced page(s).
- The page(s) is (are) not in real storage.
- The page(s) is (are) fixed.

For those pages the FCEPGOUT request will be ignored.

Return Codes in Register 15

0 All specified pages have been forced for page-out or the request has been ignored because the issuing program is running in real mode.

2 The begin address is greater than the end address, or a negative length has been found.

4 At least one of the requested pages does not belong to the partition in which the issuing program is running. The FCEPGOUT request has only been executed for those pages which belong to the partition of the issuing program.

8 a. At least one of the requested pages is temporarily fixed (via CCW-translation) and/or PFIXed. The FCEPGOUT request has only been executed for the unfixed pages.

b. A page handling request (page fault, temporary fix, PFIX) for at least one of the requested pages is pending (caused by asynchronous processing within a partition). The FCEPGOUT request has not been executed for those pages which are involved in a page handling request.

16 List of areas that are to be paged out is not completely in the requesting program's partition. The request is ignored.

Any combination of return codes 0, 2, 4, and 8 is possible.

FEOV

```
--------------------------------------------------
Name        Operation    Operand
--------------------------------------------------
[name]      FEOV         {filename|(1)}
--------------------------------------------------
```

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

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

filename|(1): The name of the file is the only parameter required. The name can be specified either as a symbol or in register notation.

FEOVD

```
--------------------------------------------------
Name        Operation    Operand
--------------------------------------------------
[name]      FEOVD        {filename|(1)}
--------------------------------------------------
```

The FEOVD (force end-of-volume for disk) macro is used for either input or output files to force an end-of-volume condition before it actually occurs. This indicates that processing of records on one volume is finished, but that more records for the same logical file are to be read from, or written on, the following volume. If extents are not available on the new volume, or if the format-1 label is posted as the last volume of the file, control is passed to the EOF address specified in the DTF.

filename|(1): The name of the file is the only required operand. The name can be specified either symbolically or in register notation.

FETCH

```
---------------------------------------------------------------
Name        Operation    Operand
---------------------------------------------------------------
[name]      FETCH        {phasename|(S,address)|(1)}
                         [,entrypoint|(S,entrypoint)|(0)]
                         [,LIST={listname|(S,listname)|(r1)}]
                         [,SYS=YES]
                         [,DE=YES]
                         [,MFG={area|(S,area)|(r2)}]
---------------------------------------------------------------
```

The FETCH macro loads and gives control to the phase specified in the first operand. If the phase is in the SVA, it is not loaded into the partition, but control is given to the phase. For information on how to load phases into the SVA and how to write SVA-eligible (reenterable) phases see VSE/Advanced Functions, System Management Guide, SC33-6094.

phasename|(S,address)|(1): For phasename specify the name of the required phase. If DE=YES is not specified, the address as specified in (S, address) or as loaded into a register points to an 8-byte field that contains the phase name. If DE=YES, the operand has a different meaning; refer to the discussion of the DE operand.

entrypoint|(S,entrypoint)|(0): Control is passed to the address specified by the entrypoint parameter. If this parameter is not specified or invalid, control is passed to the entrypoint determined at link-edit time.

If entrypoint is given in register notation, register 1 must not be used. You preload the register with the entrypoint address.

With S-type notation, the entrypoint is derived from base register and displacement, for example (S, offset (reg)). If, instead, a symbolic name is used for entrypoint, the macro expansion results in a V-type address constant. The entrypoint does not have to be identified by an EXTRN statement.

LIST={listname|(S,listname)|(r1)}: For listname specify the name of your local directory list generated in the partition by the GENL macro. When this operand is included, the system scans the local directory list for the required phasename before it initiates a search for this phase name in the pertinent core image library directory.

SYS=YES: If SYS=YES is specified, the system scans the system directory list (SDL) in the SVA and the system core image library before the private core image library (if a private CIL is assigned at all). If nothing is specified, the private CIL takes precedence.

DE=YES: This operand is useful if your program frequently fetches one specific phase. DE=YES is invalid if LIST is specified.

DE=YES indicates that your program contains a 38-byte field where you have placed a single directory entry (like those generated by the GENL macro). If this directory entry is active, the directory scan mechanism is bypassed; if not, the entry will be filled in by the supervisor after which it is active.

If the first operand is written as phasename (instead of S-type or register notation) a directory entry will be generated within the macro expansion. The generated directory entry will contain the phasename in the first 8 bytes. If you specify DE=YES and if you use (S, address) or register notation for the first operand, you must set aside the 38-byte field yourself and point to it via this operand. The directory entry must contain the phase name in the first 8 bytes (left-justified and padded with blanks), X'0D' at displacement X'0B', and X'00' at displacement X'10'.

MFG={area|(S,area)|(r2)}: The operand MFG is required if the program which issues the FETCH macro is to be reenterable. It specifies the address of a 64-byte dynamic storage area, that is, storage which your program obtained through a GETVIS macro. This area is required for system use during execution of the macro.

FREE

```
-----------------------------------------------------
Name         Operation    Operand
-----------------------------------------------------
[name]       FREE         {filename|(1)}
-----------------------------------------------------
```

The FREE macro, used in conjunction with the HOLD=YES option of a DTFxx macro, frees a portion of a DASD device that is being held under DASD record (track) protection. On a CKD device, that protected portion is a track; on an FBA device, it is an integral number of contiguous FBA blocks. On an FBA device, or a DTFSD or DTFDI file assigned to a CKD disk, the FREE macro is treated as a null operation; all holding and freeing of FBA block ranges or CKD tracks for DTFSD and DTFDI are performed implicitly by LIOCS.

The same track (or blocks) can be held more than once without an intervening FREE if the hold requests are from the same task. The same number of FREEs must be issued before the track (or block) is completely freed. However, a task is terminated if more than 16 hold requests are recorded without an intervening FREE, or if a FREE is issued to a file that does not have a hold request for that track (or block). For situations that require the use of the FREE macro, refer to VSE/Advanced Functions Application Programming: Macro User's Guide.

filename|(1): This operand is the same as the name specified in the DTF header entry.

FREEVIS

Name	Operation	Operand
[name]	FREEVIS	[ADDRESS={name1|(1)}] [,LENGTH={name2|(0)}] [,SVA={YES|<u>NO</u>}]

The FREEVIS macro releases a block (or blocks) of virtual storage that was obtained by the GETVIS macro.

If you code the macro without any operand, the system assumes that the start address of the block to be released is contained in register 1 and that the length of this block was placed into register 0. If the macro is issued without an operand, the macro must not contain a comment unless the comment begins with a comma.

ADDRESS={name1|(1)}: The start address of the virtual storage block to be released in the GETVIS area may be specified either in a 4-byte field addressed by name1 or in a register.

LENGTH={name2|(0)}: The length of the virtual storage block to be released may be specified in a 4-byte field addressed by name2 or in a register. The length is specified in number of bytes. The smallest unit of virtual storage that can be released by FREEVIS is (a) 128 bytes if the GETVIS area is part of a partition or (b) 16 bytes if the GETVIS area is part of the SVA. If the specified length is not a multiple of 128 or 16, respectively, it is rounded to the next higher integral multiple of 128 or 16.

SVA={YES|NO}: SVA=YES can be specified only in a program that is executed with storage protection key zero. If SVA=YES is specified, the system tries to find the block that is to be released in the SVA, otherwise in the pertinent partition.

Return Codes in Register 15

0	FREEVIS completed successfully.
4	The size of the (real) partition GETVIS area is OK.
8	The specified length is smaller than zero.
12	The specified address is not within the GETVIS area or the address is not (a) a multiple of 128 bytes if the GETVIS area is part of a partition, or (b) a multiple of 16 bytes if the GETVIS area is in the SVA.
16	The specified storage block to be released (ADDRESS+LENGTH) exceeds the GETVIS area.
20	Invalid FREEVIS option.

GENDTL

```
--------------------------------------------------------
Name        Operation    Operand
--------------------------------------------------------
[name]      GENDTL       [ADDR={name1|(S,name1)|(r1)}]
                         [,CONTROL={E|S}]
                         [,KEEP={NO|YES}]
                         [,LENGTH={NO|YES}]
                         [,LOCKOPT={1|2}]
                         [,NAME={name2|(S,name2)|(r2)}]
                         [,OWNER={TASK|PARTITION}]
                         [,SCOPE={INT|EXT}]
--------------------------------------------------------
```

The GENDTL macro generates a control block which is used by the LOCK/UNLOCK macros to enqueue/dequeue a resource access request. The control block, commonly called 'DTL' (Define The Lock), is generated at the time of program execution. Space for the DTL is either provided by the program, or it is to be acquired by the system.

ADDR={name1|(S,name1)|(r1)}: Specifies the address of the area where the DTL is to be built. If this operand is omitted, storage is allocated in the partition's GETVIS area by an implicit GETVIS request. After a successful GETVIS, the system places the DTL's address in register 1. Register 15 contains the return code set by the implicit GETVIS request.

CONTROL={E|S}: Defines how the named resource can be shared while your program owns it, which is determined by this specification and your specification for the operand LOCKOPT. A specification of E means the resource is enqueued for exclusive use; a specification of S means the resource is enqueued as sharable.

KEEP={NO|YES}: This operand may be used to lock the named resource beyond job step boundaries. Only a main task should use this operand. KEEP=NO indicates that the named resource once locked, is to be released automatically at the end of the particular job step. With KEEP=YES, a named resource that is locked remains locked across job steps; it will be automatically released at end-of-job.

If a job terminates abnormally, all resources with KEEP=YES are unlocked by the abnormal termination routine.

LENGTH={NO|YES}: If LENGTH=YES is specified, the GENDTL macro returns the length of the DTL in register 0. With LENGTH=NO, register 0 remains unchanged.

LOCKOPT={1|2}: This operand, together with the CONTROL parameter determines how the system controls shared access in response to a LOCK request:

- LOCKOPT=1 and CONTROL=E: No other task is allowed to use the resource concurrently.
- LOCKOPT=1 and CONTROL=S: Other 'S' users are allowed concurrent access, but no concurrent 'E' user is allowed.
- LOCKOPT=2 and CONTROL=E: No other 'E' user gets concurrent access; however, other 'S' users can have access to the resource.
- LOCKOPT=2 and CONTROL=S: Other 'S' users can have concurrent access and, in addition, one 'E' user is allowed.

All users of a particular resource have to use the same LOCKOPT specification when they lock the resource. (Exception: If LOCKOPT=1 and CONTROL=E, the lock status may be modified.)

NAME={name2|(S,name2)|(r2)}: Specifies the address of the area where a 12-byte long resource name is stored. If the name is shorter than 12 bytes, it must be padded with blanks. It is by this name that VSE controls shared access of the resource as requested by active tasks via the LOCK macro. These tasks may all be active in one partition, or they may be distributed over several partitions; the resource-share control extends across partitions.

Note that the name must not begin with any of the characters A through I or V, since these characters are reserved for IBM usage.

OWNER={TASK|PARTITION}: Defines whether the named resource, once locked, can be unlocked only by the task which issued the corresponding LOCK request (OWNER=TASK), or whether it can be unlocked by any task within the partition (OWNER=PARTITION).

When OWNER is defined as PARTITION, a LOCK request for the resource must not specify FAIL=WAIT or FAIL=WAITC because deadlock prevention (return code 16) is not supported with OWNER=PARTITION.

SCOPE={INT|EXT}: This operand may be used for locking resources across systems. SCOPE=EXT specifies that the lock is used across systems. You may omit the parameter if you want to lock your resources only on one system since the default is SCOPE=INT (that is, the locking applies to one system only).

GENIORB

```
-------------------------------------------------------
Name         Operation     Operand
-------------------------------------------------------
[name]       GENIORB       CCW={name1|(S,name1)|(r1)}
                           ,{DEVICE=SYSxxx|
                           LOGUNIT={name2|(S,name2)|(r2)}}
                           [,ADDRESS={name3|(S,name3)|(r3)}]
                           [,LENGTH=fieldlength]
                           [,ECB={name4|(S,name4)|(r4)}]
                           [,ERREXIT={name5|(S,name5)|(r5)}]
                           [,FIXLIST={name6|(S,name6)|(r6)}]
                           [,FIXFLAG=(option,...)]
                           [,IOFLAG=(option,...)]
-------------------------------------------------------
```

The GENIORB macro generates an IORB (Input/Output Request Block). The block is generated at the time of program execution. For the layout and contents of an IORB, see Figure 23 on page 170 . The IORB is an alternative to the CCB; instead of specifying a CCB in the EXCP macro, the address of an IORB is given.

The IORB requires the specification of areas to be page-fixed for the I/O operation. Such areas include the IORB and the channel programs themselves and all input/output areas. Specifying those areas frees the page-fixing routines from having to scan the channel programs to determine which areas are to be fixed.

After execution of the macro:

- register 1 contains the address of the IORB, and
- register 15 contains the return code from an implicit GETVIS.

For a detailed display in your assembly, showing the IORB fields and their meaning, issue the IORB macro with the (only) operand DSECT=YES.

CCW={name1|(S,name1)|(r1)}: This operand gives the name of the first CCW used with the IORB. The name must be the same as the name specified in the assembler CCW statement that builds the CCW.

DEVICE=SYSxxx: This operand specifies the logical unit for the actual I/O unit with which the IORB is associated.

LOGUNIT={name2|(S,name2)|(r2)}: This operand describes the device in logical unit format. It points to a halfword with the same format as a logical unit number (bytes 6 and 7) in a CCB; see Figure 2 on page 13 provided in context with the discussion of the CCB macro.

ADDRESS={name3|(S,name3)|(r3)}: If specified, this operand gives the name of the area in which the IORB is to be generated. If the ADDRESS operand is specified, the LENGTH operand should be specified as well.

Omitting the ADDRESS operand indicates that the required area is to be obtained through an implicit GETVIS issued by the system.

LENGTH=fieldlength: This operand gives the length of the field provided for IORB generation. The value must be given as a self-defining term. If this operand is omitted, a default value equal to the length of the IORB will be used; however, the assembler issues an MNOTE. If the ADDRESS operand is omitted, LENGTH is not used.

ECB={name4|(S,name4)|(r4)}: This operand specifies the address of the ECB to be posted when I/O is complete. For a more detailed description of the ECB operand, refer to the IORB macro.

ERREXIT={name5|(S,name5)|(r5)}: ERREXIT is the address of a routine to be executed should the system be unable to obtain the required virtual storage. If the ERREXIT operand is omitted, failure to obtain virtual storage causes the system to cancel the program (task).

FIXLIST, FIXFLAG, IOFLAG: For a description of those operands, refer to the IORB macro.

```
-----------------------------------------------------------
Name        Operation    Operand
-----------------------------------------------------------
[name]      GENL         phasename1,phasename2,...
                         [,ADDRESS={area|(S,area)|(r1)}
                          ,LENGTH=number]
                         [,ADDRESS={DYNAMIC|DYN}
                           [,ERREXIT={addr|(S,addr)|(r2)}]]
-----------------------------------------------------------
```

The GENL macro generates a local directory in the partition. It saves access time if you load the same phases more than once in the course of program execution.

The generated directory entries are 38 bytes long and have the following structure: The first 8 bytes contain the phase name (left-adjusted and padded with blanks if necessary); the next 4 bytes contain X'0000000D'; the remaining bytes contain X'00'.

phasename1,phasename2,...: Specify, for these parameters, the names of phases, for which entries in a local directory list are to be built. The list will be generated in alphameric sequence. You may specify up to 200 phase names, but no more than a total of 200 operands.

ADDRESS={area|(S,area)|(r1)},LENGTH=number: If the ADDRESS operand is omitted, the assembler builds a 38-byte entry within the macro expansion for each of the specified phases and inserts the pertinent phase name in the entry. The rest of the entry is filled in by the supervisor when the phase is called by a FETCH or LOAD macro, with the LIST option for the first time. When, subsequently, the phase is called again, the entry is active.

Coding ADDRESS in conjunction with LENGTH indicates that the directory is to be built, during execution, at a location within your program whose address is given by the ADDRESS operand.

LENGTH gives the length of the field provided for the generation of the directory. If LENGTH is too short the assembler issues an MNOTE.

ADDRESS={DYNAMIC|DYN}[,ERREXIT={addr|(S,addr)|(r2)}]: Coding ADDRESS=DYNAMIC (a short form, ADDRESS=DYN, is allowed) directs the system to acquire, through a GETVIS, as much dynamic storage as needed. Note that in this case the contents of registers 0, 1, 14, and 15 will be over-written by execution of the macro.

ERREXIT is the address of a routine to be executed should the implicit GETVIS fail. If the ERREXIT operand is omitted, an unsuccessful GETVIS will cause the task to be canceled.

```
------------------------------------------------------
Name          Operation    Operand
------------------------------------------------------
[name]        GET          {filename|(1)}
                           [,workname|,(0)]
------------------------------------------------------
```

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

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

filename|(1): This operand must be the same as the name of the DTF macro for the file from which the record is to be retrieved. The filename can be specified as a symbol or in (special or ordinary) register notation (to make your program self-relocating). The high-order bits of the register must be zero, or unpredictable results may occur.

workname|(0): This operand specifies a work area name or a register (in either special or ordinary register notation) containing the address of the work area. The work area address should never be preloaded into register 1.

The operand is used if records are to be processed in a work area which you define yourself (for example, using a DS instruction). Specifying a work area causes GET to move each individual record from the input area to the work area. If the operand is specified, all GETs for the named file must use either a register or a workname.

The workname operand is not valid for the 3881. Also, you cannot specify the WORKA operand in the DTFCD for the 3881.

In conjunction with optical reader input, GET can be used only to retrieve records from a journal tape on a 1287.

GETIME

```
------------------------------------------------------------
Name          Operation     Operand
------------------------------------------------------------
[name]        GETIME        [STANDARD|BINARY|TU|MIC]
                            [,LOCAL|,GMT]
                            [,CLOCK=YES]
                            [,MFG={area|(S,area)|(r)}]
------------------------------------------------------------
```

The GETIME macro obtains the time-of-day at any time during program execution.

STANDARD and LOCAL are assumed if no operands are given. If the macro is issued without an operand, the macro must not contain a comment unless the comment begins with a comma.

As long as no DATE job control statement is supplied, the calendar date and the system date in the communication region are updated every time GETIME is issued. Those dates are therefore accurate at any given moment. However, when the job stream contains a DATE job control statement, only the system date in the communication region is updated when GETIME is used; the calendar date is not changed in that case.

STANDARD|BINARY|TU|MIC: If STANDARD is specified, the time-of-day is returned in register 1 as a packed decimal number of the form hhmmss, where hh is hours, mm is minutes, and ss seconds, with the sign in the low-order half-byte. The time-of-day may be stored and unpacked or edited.

If BINARY is specified, the time-of-day is returned in register 1 as a binary integer in seconds.

If TU is specified, the time-of-day is returned in register 1 as a binary integer in units of 1/300 seconds.

If MIC is specified, the time-of-day is returned in registers 0 and 1 as a binary integer in microseconds. Bit 51 of the register pair indicates one microsecond.

LOCAL|GMT: Specify LOCAL to obtain the local time or GMT if, in your program, you want to use Greenwich Mean Time.

CLOCK=YES: Indicates that registers 0 and 1 contain, as input to GETIME, a value that was obtained by means of a STCK (store clock) instruction. This value is transformed into time and date as defined by the other operands and any associated job control statements. On output, register 1 contains the time, and registers 14 and 15 the date (in the form mmddyy00 or ddmmyy00).

MFG={area|(S,area)|(r)}: The MFG operand is required if the program is to be reenterable and if option STANDARD applies (with the options BINARY, TU, or MIC, reentrancy is preserved in any case). MFG specifies the address of a 64-byte dynamic storage area, that is, storage which your program obtained through a GETVIS macro. This area is required for system use during execution of the macro.

GETVIS

```
---------------------------------------------------
Name        Operation    Operand
---------------------------------------------------
[name]      GETVIS       [ADDRESS={name1|(1)}]
                         [,LENGTH={name2|(0)}]
                         [,PAGE=YES]
                         [,POOL=YES]
                         [,SVA=YES]
---------------------------------------------------
```

The GETVIS macro retrieves a block of virtual storage from the GETVIS area of your partition or of the SVA. If you code the macro without any operand, the system assumes that the length of the desired virtual storage area is contained in register 0 and returns the start address of the area it retrieved for you in register 1. If the macro is issued without an operand, the macro must not contain a comment unless the comment begins with a comma.

ADDRESS={name1|(1)}: The start address of the requested virtual storage area is returned by the system either in the 4-byte field specified as a symbol by name1 or in the specified register. (Register 15 must not be used because it contains the return code.) The returned address is valid only if the return code in register 15 is zero. If the operand is omitted, the address is returned in register 1 only.

LENGTH={name2|(0)}: The length of the requested storage block may be specified either in the 4-byte field (specified as a symbol by name2) or in the specified register. The length is specified in number of bytes. The smallest unit that can be requested by GETVIS is (a) 128 bytes if the GETVIS area is part of a partition or (b) 16 bytes if the GETVIS area is part of the SVA. If the specified length is not a multiple of 128 or 16, respectively, it is rounded to the next higher multiple of 128 or 16. If the operand is omitted, the system assumes that you have specified the length in register 0.

PAGE=YES: If you want the requested storage area to start on a page boundary, specify PAGE=YES. This may reduce the number of page faults.

POOL=YES: If POOL=YES is specified, GETVIS starts searching for the requested virtual storage area at the address specified in register 1. In this case, it is your responsibility to provide a valid address in register 1.

SVA=YES: SVA=YES can be specified only in a program that is executed with storage protection key zero. If SVA=YES is specified, the system retrieves the desired block of virtual storage from the SVA. Otherwise, it retrieves the block from the pertinent partition.

GETVIS

Return Codes in Register 15

0	GETVIS completed successfully.
4	The size of the (real) partition GETVIS area is OK.
8	The specified length is negative or exceeds the GETVIS area.
12	No more virtual storage is available in the GETVIS area.
20	Invalid GETVIS option.
32	A hardware (storage) failure occurred in the requested real partition GETVIS area.

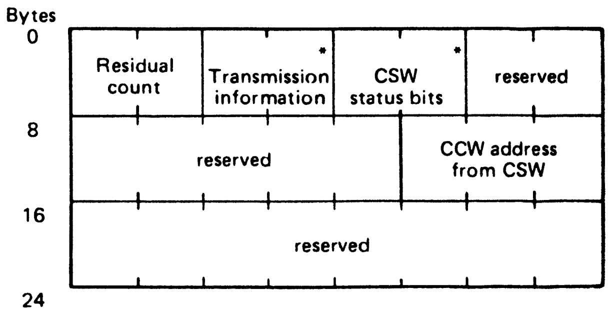
IJBPUB

```
------------------------------------------------------
Name        Operation    Operand
------------------------------------------------------
[name]      IJBPUB
------------------------------------------------------
```

The IJBPUB macro generates a mapping DSECT, which is used to interpret the information retrieved with the EXTRACT ID=PUB macro. The channel and device number are contained in the first two bytes. Device type code and the device characteristic code are stored in bytes five and six, respectively.

```
------------------------------------------------------------
Name          Operation     Operand
------------------------------------------------------------
[name]        IORB          DSECT=YES
                            or
                            CCW=name1,DEVICE=SYSxxx
                            [,ECB=name2]
                            [,FIXLIST=name3]
                            [,FIXFLAG=(option,...)]
                            [,IOFLAG=(option,...)]
------------------------------------------------------------
```

The IORB macro generates an IORB (Input/Output Request Block). The block is generated when your program is being assembled. For the layout and contents of an IORB, see Figure 23 .



* Same as for a CCB (see Figure 4-1)

Figure 23. Layout and Contents of the I/O Request Block (IORB)

The IORB is an alternative to the CCB: instead of specifying a CCB in the EXCP macro, the address of an IORB is given. The IORB macro requires the specification of areas to be page-fixed for the I/O operation. Such areas include the IORB and the channel programs themselves and all input/output areas. Specifying those areas frees the page-fixing routines from having to scan the channel programs to determine which areas are to be fixed.

DSECT=YES: If the operand is specified, it should be the only one. Any other parameters specified in the macro are ignored and an appropriate MNOTE is generated by the assembler.

Specifying DSECT=YES causes the assembler to display, as a DSECT structure, the IORB and the meaning of its fields.

CCW=name1: This operand gives the name of the first CCW used with the IORB. This name must be the same as the name specified in the assembler CCW statement that builds the CCW.

DEVICE=SYSxxx: This operand specifies the logical unit for the actual I/O unit with which this IORB is associated.

ECB=name2: This operand specifies the address of the ECB to be posted when I/O is complete. The traffic bit (byte 2, bit 0) of the ECB must have been cleared before issuing the EXCP macro. The ECB area must be included in the fixlist if the ECB operand is used.

> **Note:** If FIXFLAG=(FIXED) is specified, the ECB must have been PFIXed.

FIXLIST=name3: This operand specifies the address of the first of two or more fixlist parts or of the only fixlist part. The FIXLIST operand is required unless FIXFLAG=(FIXED) is specified. Each fixlist part consists of one or more 8-byte entries plus an end or chaining indicator as shown in Figure 24 .

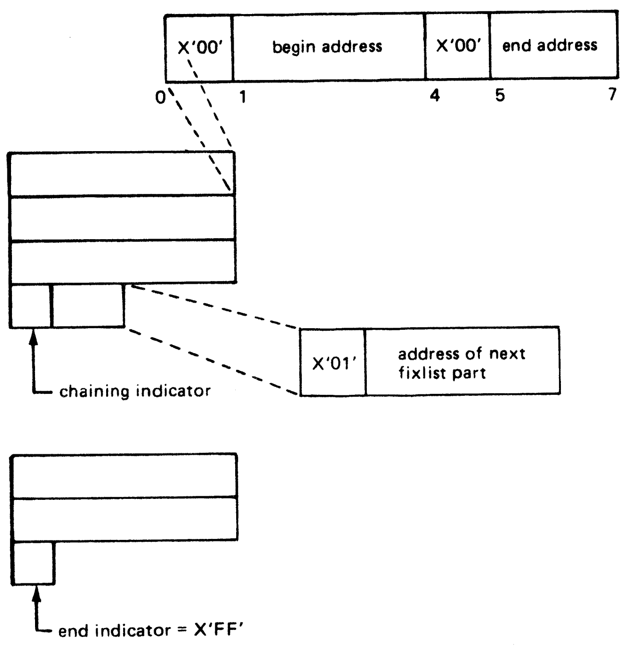


Figure 24. Layout of Fixlist

In a fixlist entry, begin address and end address are the addresses of the first and the last byte of an area which has to be fixed for the I/O request (begin address ≤ end address; entries with begin address > end address will be canceled with an 'invalid address' message). Each entry describes a storage area that is accessed by the channel during the I/O request; that is, an area containing the channel program, an input/output area, or the IORB.

Duplicate entries and entries describing overlapping storage areas are allowed. As a result, certain areas may be covered more than once by the fixlist. The system will compress the fixlist so that each page to be fixed for the channel program is covered only once. However, specifying FIXFLAG=(COMPRESSED) indicates that this service is not applicable or has already been done by the user.

FIXFLAG=(option,...): This operand specifies a list of options which apply to the I/O fixing procedure. In S/370 mode, this specification is ignored.

The options you can specify are:

COMPRESSED	to indicate that the system does not have to compress the fixlist. Use this option if the fixlist is already compressed, that is: each page to be fixed for the I/O request is covered only once by the fixlist.
FIXED	to indicate that all areas which should be fixed for the I/O operation have already been fixed by the user, there is no fixlist.

IOFLAG=(option,...): A list of options may be specified which apply to I/O interrupt handling:

POSTDE	to indicate that device end is to be posted.
POSTERR	to indicate that an unrecoverable I/O error is to be accepted.
SKIPERP	to indicate that error recovery by the system is to be skipped.

ISMOD

```
-----------------------------------------------------------------
Name        Operation    Operand
-----------------------------------------------------------------
[name]      ISMOD        IOROUT={LOAD|ADD|RETRVE|ADDRTR}
                         [,CORDATA=YES]
                         [,CORINDX=YES]
                         [,ERREXT=YES]
                         [,HOLD=YES]
                         [,IOAREA2=YES]
                         [,RDONLY=YES]
                         [,RECFORM={FIXUNB|FIXBLK|BOTH}]
                         [,RPS=SVA]
                         [,SEPASMB=YES]
                         [,TYPEFLE={RANDOM|SEQNTL|RANSEQ}]
-----------------------------------------------------------------
```

The ISMOD macro defines a logic module for an ISAM file. If you do not provide a name for the module, IOCS generates a standard module name.

> **Note:** If an ISMOD module precedes an assembler language USING statement or follows your program, registers 2-12 remain unrestricted even at assembly time. However, if the ISMOD module lies within your program, you should issue the same USING statement (as that which was issued before the ISMOD module) directly following the module. This action is necessary because the ISMOD module uses registers 1, 2, and 3 as base registers, and the ISMOD CORDATA module uses registers 1, 2, 3, and 5 as base registers. Each time either module is assembled, these registers are dropped.

CORDATA=YES: Include this operand if the module is to add records to files with the IOSIZE DTFIS operand. If this operand is included, the IOSIZE operand is required in the DTF. If you omit the CORDATA=YES operand, you will not have an increase in throughput when adding records to a file.

CORINDX=YES: Include this operand to generate a module that can process DTFIS files (add or random retrieve functions) with or without the cylinder index entries resident in virtual storage. If omitted, the module generated cannot process the resident cylinder index entries.

If an unrecoverable I/O error occurs while reading indexes into virtual storage, the program will not use the resident cylinder index entries.

ERREXT=YES: Include this operand if the ERET macro is to be used with this module or if non-data-transfer error conditions are returned in filenameC. If HOLD=YES and ERREXT=YES, your program must issue the ERET macro to return to the ISAM module to free any held tracks. See the DTF ERREXT and HOLD operands.

HOLD=YES: This operand provides for the track hold option for both data and index records. If the HOLD operand is omitted, the track hold function is not performed.

Because track hold cannot be performed on a LOAD file, HOLD=YES cannot be specified when IOROUT=LOAD.

If HOLD=YES and ERREXT=YES, your program must issue the ERET macro to return to the ISAM module to free any held tracks.

IOAREA2=YES: Include this operand if a second I/O area is to be used, that is, if IOAREA2 is specified in any of the DTFs linked to the logic module. The operand is only valid for load or sequential retrieval functions. The module can process DTFs with one or two I/O areas specified. This operand must not be specified if TYPEFLE=RANSEQ is specified.

IOROUT={LOAD|ADD|RETRVE|ADDRTR}: This operand specifies the type of module required to perform a given function.

LOAD generates a module for creating or extending a file.

ADD generates a module for adding new records to an existing file.

RETRVE generates a module to retrieve, either randomly or sequentially, records from a file.

ADDRTR generates a module that combines the features of the ADD and RETRVE modules. This module also processes any file in which only ADD or RETRVE is specified in the IOROUT operand of the DTF, and in which the TYPEFLE operand contains the corresponding parameter (or a subset of it).

RDONLY=YES: This operand causes a read-only module to be generated. Whenever this operand is specified, any DTF used with the module must have the same operand.

RECFORM={FIXUNB|FIXBLK|BOTH}: This operand generates a module that creates, adds to, or processes an unblocked (FIXUNB) or blocked (FIXBLK) file. If BOTH is specified, a module is generated to process both unblocked and blocked files, and the DTF may specify either FIXUNB or FIXBLK in the RECFORM operand. The RECFORM operand is required only when IOROUT specifies ADD or ADDRTR. If IOROUT specifies LOAD or RETRVE, a module that handles fixed-length blocked and unblocked files is generated, and the operand is not required.

RPS=SVA: This operand causes the RPS logic modules to be assembled.

SEPASMB=YES: Include this operand only if the module is to be assembled separately. This causes a CATALR card with the filename to be punched ahead of the object deck and the filename to be defined as an ENTRY point in the assembly. If the operand is omitted, the assembler assumes that the DTF is assembled together with the problem program.

TYPEFLE={RANDOM|SEQNTL|RANSEQ}: This operand is required when IOROUT specifies RETRVE or ADDRTR. RANDOM generates a module that includes only random retrieval capabilities. SEQNTL generates a module that includes only sequential retrieval capabilities. RANSEQ generates a module that includes random and sequential capabilities. It also processes any file in which the TYPEFLE operand specifies either RANDOM or SEQNTL. If TYPEFLE=RANSEQ, IOAREA2=YES must not be specified.

When all operands are omitted, the ISMOD module can only process files where IOROUT=RETRVE, TYPEFLE=RANSEQ, CORINDX, CORDATA, HOLD, and RDONLY are not specified. The name of that module is IJHZRBZZ.

Standard ISMOD Names

Each name begins with a 3-character prefix (IJH) and continues with a 5-character field corresponding to the options permitted in the generation of the module.

ISMOD name = IJHabcde

a = A	RECFORM=BOTH, IOROUT=ADD or ADDRTR
= B	RECFORM=FIXBLK, IOROUT=ADD or ADDRTR
= U	RECFORM=FIXUNB, IOROUT=ADD or ADDRTR
= Z	RECFORM is not specified. (IOROUT=LOAD or RETRVE)

b = A	IOROUT=ADDRTR
= I	IOROUT=ADD
= L	IOROUT=LOAD
= R	IOROUT=RETRVE
= V	IOROUT=ADDRTR, RPS=SVA
= X	IOROUT=LOAD, RPS=SVA

c = B	TYPEFLE=RANSEQ
= G	IOAREA2=YES, TYPEFLE=SEQNTL or IOROUT=LOAD
= R	TYPEFLE=RANDOM
= S	TYPEFLE=SEQNTL
= Z	neither is specified (IOROUT=LOAD or ADD)

d = B	CORINDX=YES and HOLD=YES
= C	CORINDX=YES
= O	HOLD=YES
= Z	neither is specified

e = F	CORDATA=YES, ERREXT=YES, RDONLY=YES
= G	CORDATA=YES and ERREXT=YES
= O	CORDATA=YES and RDONLY=YES
= P	CORDATA=YES
= S	ERREXT=YES and RDONLY=YES

= T	ERREXT=YES
= Y	RDONLY=YES
= Z	neither is specified

Subset/Superset ISMOD Names

The following chart shows the subsetting and supersetting allowed for ISMOD names. Five parameters allow supersetting. For example, the module IJHBABZZ is a superset of the module IJHBASZZ.

```
-------------------------------------------
          + + + + +
    I J H A A B B F
          B I R O O
          Z + + + +
          + A B C S
          A R S Z Y
          U * +   +
          Z L G   G
              S   P
              +   +
              G   T
              Z   Z
-------------------------------------------
   + Subsetting/supersetting permitted.
   * No subsetting/supersetting permitted.
-------------------------------------------
```

If two or more modules with the same entry point are included, the linkage editor message 2143I, (invalid duplication of entry point label) is generated. (Occasionally these entry points are not obvious when using the preceding chart, but the module can perform the indicated functions.) This message can usually be suppressed by including a superset module. However, modules with and without prime data in main storage or modules with TYPEFLE=RANDOM and IOAREA2=YES cannot be combined. Therefore, you should take either of the following actions:

1. Specify prime data in core for each ADD type DTF in your program. In this case, superset modules are generated.

2. Specify the MODNAME operand in the DTF, and include an ISMOD of that name. The DTF then generates only the specified module.

JDUMP

```
--------------------------------------------------
Name         Operation    Operand
--------------------------------------------------
[name]       JDUMP
--------------------------------------------------
```

This macro provides a hexadecimal dump of the following:

- The contents of either the entire supervisor area and the used part of the system GETVIS area, or of some supervisor control blocks only (see Note below).
- The contents of the partition that issued the macro.
- The contents of the registers.

Note: The dump includes the contents of some supervisor control blocks only, rather than the entire supervisor area, if the STDOPT job control command specifies DUMP=PART or NO, or if a job control statement // OPTION PARTDUMP or NODUMP is submitted.

In addition, the macro causes the job to be terminated if JDUMP was issued by the main (or only) task of the program. If JDUMP was issued by a subtask, the macro causes that subtask to be detached without terminating the program in the partition.

The dump provided by the macro is always directed to SYSLST; SYSLST, if disk or tape, must be opened; if SYSLST is a tape, that tape must be positioned as desired.

If JDUMP is issued by a job running in real mode, the storage contents of the partition are dumped only up to the limit as determined by the SIZE parameter of the EXEC job control statement, plus the storage obtained dynamically through the GETVIS macro. If SIZE was not specified, the entire partition will be dumped.

If JDUMP is issued by a program running in virtual mode, the entire partition is dumped.

JOBCOM

```
--------------------------------------------------------
Name          Operation     Operand
--------------------------------------------------------
[name]        JOBCOM        FUNCT={PUTCOM|GETCOM},
                            AREA={address|(r1)},
                            LENGTH={length|(r2)}
--------------------------------------------------------
```

The JOBCOM macro allows for communication between jobs or job steps in a partition. Information being communicated is stored in a 256-byte area. The system provides such an area for each partition. Through the JOBCOM macro, a program either moves information into that area or retrieves information that had previously been stored there by another program. The area remains unaltered from one job (or job step) to the next. Unless it is modified through execution of the JOBCOM macro, the contents of the area remain unchanged over any number of jobs. The JOBCOM macro is not reentrant.

The program that issues the JOBCOM macro must provide a register save area 18 fullwords long. Prior to execution of the macro, register 13 has to point to that save area.

Note: When the JOBCOM macro is used, registers 1 through 14 are destroyed.

FUNCT={PUTCOM|GETCOM}: This operand describes the function that the macro is to perform. Specifying PUTCOM causes information to be stored into the system-supplied area. The number of bytes to be moved is given by the LENGTH operand. If LENGTH yields a value smaller than 256, the remainder of the area is left unaltered.

Specifying GETCOM indicates that information is to be retrieved from the system-supplied area. Again, the number of bytes to be moved is given by the LENGTH operand.

AREA={address|(r1)}: This operand gives the address of a field where the program provides (FUNCT=PUTCOM specified) or receives (FUNCT=GETCOM specified) the information to be moved.

LENGTH={length|(r2)}: This operand specifies the number of bytes to be moved. The value is either given as a self-defining term or in register notation. If register notation is used, the specified register is expected to contain the length value.

Length should be a positive number up to 256. If it is zero or negative, no information gets moved. If it is greater than 256, only 256 bytes are moved.

LBRET

```
-------------------------------------------------------
Name          Operation    Operand
-------------------------------------------------------
[name]        LBRET        {1|2|3}
-------------------------------------------------------
```

The LBRET macro is issued in your subroutines when you have completed processing labels and wish to return control to IOCS. LBRET applies to subroutines that write or check DASD or magnetic tape user-standard labels, write or check tape nonstandard labels, or check DASD extents. The operand used - 1, 2, or 3 - depends on the function to be performed. The functions and operands are explained below.

Checking DASD Extents: When processing an input file with all volumes mounted, you can process your extent information. After each extent is processed, use LBRET 2 to receive the next extent. When extent processing is complete, use LBRET 1 to return control to IOCS.

Checking User Standard DASD Labels: IOCS passes the labels to you one at a time until the maximum allowable number is read (and updated), or until you signify you want no more. In the label routine, use LBRET 3 if you want IOCS to update (rewrite) the label just read and pass you the next label. Use LBRET 2 if you simply want IOCS to read and pass the next label. If an end-of-file record is read when LBRET 2 or LBRET 3 is used, label checking is automatically ended. If you want to eliminate the checking of one or more remaining labels, use LBRET 1.

Writing User Standard DASD Labels: Build the labels one at a time and use LBRET to return to IOCS, which writes the labels. Use LBRET 2 if you want control returned to you after IOCS writes the label. If, however, IOCS determines that the maximum number of labels has already been written, label processing is terminated. Use LBRET 1 if you wish to stop writing labels before the maximum number of labels is written.

Checking User Standard Tape Labels: IOCS reads and passes the labels to you one at a time until a tapemark is read, or until you indicate that you do not want any more labels. Use LBRET 2 if you want to process the next label. If IOCS reads a tapemark, label processing is automatically terminated. Use LBRET 1 if you want to bypass any remaining labels.

Writing User Standard Tape Labels: Build the labels one at a time and return to IOCS, which writes the labels. When LBRET 2 is used, IOCS returns control to you (at the address specified in the DTFxx LABADDR operand) after writing the label. Use LBRET 1 to terminate the label set.

Writing or Checking Nonstandard Tape Labels: You must process all your nonstandard labels at once. Use LBRET 2 after all label processing is completed and you want to return control to IOCS.

LFCB

Name	Operation	Operand
[name]	LFCB	SYSxxx,phasename [,NULMSG] [,FORMS=xxxx] [,LPI=n]

The macro can be used to load the forms control buffer (FCB) of a printer dynamically. That printer must not be an IBM 3800 Printing Subsystem; the macro is ignored on an IBM 3800. An FCB whose contents have been changed by means of this macro retains the changed contents until one of the following occurs:

- another LFCB macro is issued for the printer
- an LFCB command is issued for the printer
- the SYSBUFLD program is executed to reload the printer's FCB
- IPL is performed for the system.

The macro, when executed, generates messages to request operator action (such as changing forms), whenever manual action is required, and to inform the operator that the FCB of the specified printer has been reloaded.

> **Note:** If SYSLOG is assigned to the printer, the results of an FCB load operation initiated by an LFCB macro are unpredictable.

SYSxxx: The name of the logical unit associated with the printer whose FCB is to be loaded.

You can specify one of the following:

- SYSLST
- SYSLOG
- SYSnnn, a programmer logical unit assigned to a printer owned by the partition in which the program is executed.

phasename: The name by which the phase containing the applicable FCB image is cataloged in the core image library. For information on the contents and format of an FCB image, see VSE/Advanced Functions, System Control Statements, SC33-6095.

NULMSG: This operand specifies that the 80-character verification message, which is normally printed following the FCB load operation, is to be suppressed. This operand, if given, must be specified immediately after phasename.

If this operand is specified, the system continues normal processing immediately after the FCB load operation has been completed, and the operator cannot verify that the proper forms are placed on the printer.

If the operand is omitted, the system prints the last 80 characters of the phase identified by phasename, and positions the printer to the first printable line on the forms.

FORMS=xxxx: This operand specifies the type of forms to be used on the printer whose FCB is being reloaded. For xxxx, a string of up to four alphameric characters can be specified. The specified form number is included in a message to the operator.

LPI=n: This operand, which should not be given for a PRT1-printer (with a device type code of PRT1), specifies the desired number of lines per inch. For n, you can specify either 6 or 8.

If the macro is issued for a PRT1-printer and the specified spacing disagrees with the spacing code in the new buffer image, the system does not execute the FCB load operation and sets the appropriate return code in register 15.

If the macro is issued for a non-PRT1-printer, the system includes the operand in a message to the operator.

Return Codes in Register 15

Successful completion of the FCB load operation is indicated to the problem program by a return code of 0. Note, however, that for an IBM 3800, register 15 contains 0, although the macro was not executed. If the operation fails, register 15 contains one of the return codes listed below; in this case the FCB retains its original contents. The return codes are:

Dec	Hex	Meaning
4	X'04'	The assigned printer is a PRT1-printer and the LPI operand specified in the macro disagrees with the FCB image.
8	X'08'	No LUB is available for the specified logical unit.
12	X'0C'	The specified logical unit has not been assigned or is currently unassigned.
16	X'10'	The specified logical unit is assigned to a device without an FCB.

20	X'14'	The printer assigned to the specified logical unit is down.
24	X'18'	The specified FCB image phase has not been found.
28	X'1C'	The specified FCB image phase is invalid for the printer assigned to the specified logical unit.

By testing register 15, you can determine in your program whether or not the operation has failed. If the operation has failed, you can either terminate the job step or continue processing. Should you decide to continue processing, then the system bypasses the execution of the LFCB macro.

LITE

```
-----------------------------------------------------
Name         Operation    Operand
-----------------------------------------------------
[name]       LITE         {filename|(1)}
                          [,light-switches|,(0)]
-----------------------------------------------------
```

This macro lights any combination of pocket lights on a 1419 magnetic character reader or 1275 optical reader/sorter. Before using the LITE macro, the DISEN macro must be issued to disengage the device. Processing of the documents should be continued until the unit exception bit (byte 0, bit 3) of the document buffer status indicators is set on (see Figure 4 on page 31). When this bit is on, the follow-up documents have been processed, the MICR reader has been disengaged, and the pocket LITE macro can be issued.

filename|(1): Is the name of the file; this name is the same as that specified for the DTFMR header entry for the file.

light-switches|(0): Indicates a 2-byte area containing the pocket light switches. Both operands can be given either as a symbol or in register notation.

The bit configuration for the pocket light switch area is shown in Figure 25 . The pocket lights that are turned on should have their indicator bits set to 1. If an error occurs during the execution of the pocket lighting I/O commands, bit 7 in byte 1 is set to 1. This error condition normally indicates that the pocket light operation was unsuccessful.

Bits	0	1	2	3	4	5	6	7	8	9	A	B	C D E	F
Pocket Lights	A	B	0	1	2	3	4	5	6	7	8	9	Reserved	Error indicator bit

Figure 25. Bit Configuration for Pocket Light Switch Area of the 1419

LOAD

```
--------------------------------------------------------------
Name         Operation    Operand
--------------------------------------------------------------
[name]       LOAD         {phasename|(S,address)|(1)}
                          [,loadpoint|(S,loadpoint)|(0)]
                          [,LIST={listname|(S,listname)|(r1)}]
                          [,SYS=YES]
                          [,DE=YES]
                          [,TXT=NO]
                          [,MFG={area|(S,area)|(r2)}]
--------------------------------------------------------------
```

The LOAD macro loads the phase specified in the first operand (if this phase is not in the SVA) and returns control to the calling phase. After execution of the macro, the entry-point address of the called phase is returned to you in register 1. For a non-relocatable phase, this address is the entry-point determined at link-edit time. For a relocatable phase, the entry point is adjusted by the relocation factor. If the phase is in the SVA, it is not loaded; the entry point address in the SVA, however, is returned in register 1.

phasename|(S,address)|(1): For phasename specify the name of the required phase. If DE=YES is not specified, the address as specified in (S, address) or as loaded into a register points to an 8-byte field that contains the phase name. If DE=YES, the operand has a different meaning; refer to the discussion of the DE operand.

loadpoint|(S,loadpoint)|(0): If loadpoint is provided, the load-point address specified to the linkage editor is overridden, and the phase is loaded at the specified address. The address used must be outside the supervisor area. When an overriding address is given, the entry-point address is relocated and returned in register 1. An overriding loadpoint address must not be specified for a phase that had been linked as a member of an overlay structure.

If the phase is non-relocatable, none of the other addresses in the phase are relocated; if the phase is relocatable, however, the entry point and address constants are updated with the relocation factor.

If loadpoint is given in register notation, the register used must not be register 1. Preload the register with the loadpoint address.

With (S,...) notation, the loadpoint address is derived from base register and displacement as assembled for loadpoint in the (S,loadpoint) specification.

LIST={listname|(S,listname)|(r1)}: For listname specify the name of your local directory list generated in the partition by the GENL macro. When this operand is included, the system scans the local directory list for the name of the required phase before it initiates a search for this phase name in the pertinent core image library directory.

SYS=YES: If SYS=YES is specified, the system scans the system directory list (SDL) in the SVA and the system core image library before the private core image library (if a private CIL is assigned at all). If nothing is specified, the private CIL takes precedence.

DE=YES: This operand is useful if your program frequently loads one specific phase. DE=YES is invalid if LIST is specified.

DE=YES indicates that your program contains a 38-byte field where you have placed a single directory entry (like those generated by the GENL macro). If this directory entry is active, the directory scan mechanism is bypassed; if not, the entry will be filled in by the supervisor and then becomes active.

If the first operand is written as phasename (instead of S-type or register notation) a directory entry will be generated within the macro expansion. The generated directory entry will contain the phasename in the first 8 bytes.

If you use (S,address) or register notation for the first operand, you must set aside the 38-byte field yourself and point to it via this operand. The directory entry must contain the phase name in the first 8 bytes (left-justified and padded with blanks), X'0D' at displacement X'0B', and X'00' at displacement X'10'.

TXT=NO: The specification TXT=NO (with LIST=listname or DE=YES) is useful if a phase is loaded more than once in the course of your program. It causes a search for the directory entry without transfer of the contents (or text) of the phase itself and it indicates in the directory entry if and where the phase was found. This can be used to accomplish either of the following:

1. The directory entry can be filled in from the core image library for later FETCH/LOAD calls without the overhead of text transfer.

2. You can establish whether a given phase is present in a core image library or the SVA since register 0 contains the address of the directory entry and byte 16 of the directory entry is:

X'06'	if the phase is not found
X'12'	if the phase is in the SVA
X'0A'	if the phase is in the private CIL

   Note: Test for these conditions by means of a Test Under Mask (TM) instruction, not a Compare instruction. If the phase is not found and both DE=YES and TXT=NO have been specified, register 1 is returned with X'00'.

MFG={area|(S,area)|(r2)}: The operand MFG is required if the program which issues the LOAD macro is to be reenterable. It specifies the address of a 64-byte dynamic storage area, that is, storage which your program obtained through a GETVIS macro. This area is required for system use during execution of the macro.

LOCK

```
-----------------------------------------------------------
Name        Operation    Operand
-----------------------------------------------------------
[name]      LOCK         {name|(S,name)|(r)}
                         [,FAIL={RETURN|WAITC|WAIT}]
-----------------------------------------------------------
```

The LOCK macro enqueues the task for accessing the named resource. The resource must have been defined in the program by a DTL (Define The Lock) control block. A DTL is generated by issuing a DTL or GENDTL macro; it may be modified by issuing a MODDTL macro.

> **Note:** Do not LOCK a resource in an AB exit routine if this resource is held by the main task, since a deadlock situation may occur.

{name|(S,name)|(r)}: Specifies the DTL address.

FAIL={RETURN|WAITC|WAIT}: Defines the system action in case the resource cannot be obtained:

RETURN causes the system to return control back to the requesting program in any case. The requesting program has to check the return code in register 15 to find out whether or not the request was successful.

WAITC causes the system to place the requesting task in the wait state if the requested resource is found to be locked by another task. In all other cases, control returns to the requesting program. The requesting program has to check the return code in register 15 to find out whether the request was successful, or whether an error occurred.

WAIT requests the system to return control to the requesting task when the resource can be obtained. If the resource is locked by another task, the requesting task is set into the wait state until the resource is freed. In case of an error condition (return codes 12, 16, 20, 24, 32, 36), the requesting task is canceled.

WAIT or WAITC cannot be specified if the resource is defined with OWNER=PARTITION.

Return Codes in Register 15

0	Successful request: the resource is locked for the task (or for the partition if the resource is defined with partition ownership).
4	Resource not available: the resource is already locked with a locking status that allows no concurrent access.

8	The lock table space is exhausted.
12	The lock request is inconsistent with previous lock requests (by the same or other tasks).
16	The request would have resulted in a deadlock condition within the system (deadlocks across systems are not affected).
20	DTL format error.
24	The issuing task tried to lock a resource, which it owns already exclusively.
28	The lock request resulted in lock file overflow condition. Use the DLF command to specify a larger size for the lock file.
32	A lock request was issued for a shared DASD file but the corresponding volume is not on-line.
36	An unrecoverable I/O error occurred on the lock file. This probably means that the system has to be re-IPLed and the lock file re-defined. (This has to be done on all sharing systems.)

Figure 26 on page 189 presents a summary of system actions by return codes, depending on the specification of the FAIL operand.

Figure 27 on page 189 summarizes how the system controls access to a resource, depending on the specification of the CONTROL and LOCKOPT operands in the DTL or GENDTL macro. The illustration assumes that a task issues a LOCK request for a resource which is already locked.

A task or partition may lock a resource more than once. The system maintains a lock request count for the resource.

When a resource is defined with LOCKOPT=1, a task may issue up to 255 LOCK requests with CONTROL=S. When a resource is defined with LOCKOPT=2, up to 255 LOCK requests with CONTROL=S and (if no other task locks the resource exclusively) one LOCK request with CONTROL=E are allowed.

When a resource is locked more than once by a task, this task has to issue at least as many UNLOCK requests as it issued LOCK requests before it gives up the resource completely. If the resource is defined with OWNER=PARTITION, the unlocking may be done by any task in the partition.

Return Code	LOCK FAIL =		
	RETURN	WAITC	WAIT
0	RETURN	RETURN	RETURN
4	RETURN	WAIT	WAIT
8	RETURN	RETURN	WAIT
12	RETURN	RETURN	CANCEL
16	RETURN	RETURN	CANCEL
20	RETURN	RETURN	CANCEL
24	RETURN	RETURN	CANCEL
28	RETURN	RETURN	WAIT
32	RETURN	RETURN	CANCEL
36	RETURN	RETURN	CANCEL

Figure 26. System Actions by Return Code and FAIL Operand

Control definition in owning DTL		Control definition in requesting DTL			
		LOCKOPT=1		LOCKOPT=2	
		CONTROL=S	CONTROL=W	CONTROL=S	CONTROL=W
LOCKOPT=1	CONTROL=S	G	W	I	I
	CONTROL=E	W	W	W	W
LOCKOPT=2	CONTROL=S	I	W	G	G
	CONTROL=E	I	W	G	W

G	The lock request is granted (return code 0).
I	The lock request is inconsistent with current lock status (return code 12).
W	Access to the resource cannot be granted (return code 4 or 16).

Figure 27. System Actions Depending on Control Definition in DTLs

MAPBDY

Name	Operation	Operand
[name]	MAPBDY	[DSECT={NO|YES}]

The MAPBDY macro may be used to interpret the information retrieved by the EXTRACT ID=BDY macro. If 'name' is omitted, MAPBDY is taken as default.

DSECT={NO|YES}: DSECT=YES specifies that a mapping DSECT is generated. If the operand is omitted, in-line code is generated.

MAPSSID

```
----------------------------------------------------
Name          Operation    Operand
----------------------------------------------------
[name]        MAPSSID
----------------------------------------------------
```

The MAPSSID macro generates a mapping DSECT which is used to interpret the supervisor information retrieved with the SUBSID macro.

The output shows the supervisor identification string in the following format:

Displacement Dec.	Displacement Hex.	Length Byte	Contents/Description
Fixed part:			
0	0	2	Zero
2	2	4	Character string: SUP
6	6	1	Version number
7	7	1	Release number
8	8	1	Modification level
9	9	1	Length of variable part (maximum 24 bytes)
Variable part:			
10	A	1	Flag byte 1: X'80' /370 mode supervisor X'40' ECPS:VSE mode supervisor X'20' CKD support available X'10' FBA support available X'08' 3800 support available X'04' Relocating channels (ECPS:VSE–mode only) X'02' VMLE mode supervisor X'01' VMLE active under VM control
11	B	1	Flag byte 2: X'80' AF support available
12	C	1	Flag byte 3: X'80' Security support active X'40' DASD sharing support available X'20' For internal use
13	D	1	Flag byte 4: Not used
14	E	2	Library concatenation chain length

MODDTL

```
------------------------------------------------------------
Name        Operation     Operand
------------------------------------------------------------
[name]      MODDTL        ADDR={name1|(S,name1)|(r1)}
                          [,NAME={name2|(S,name2)|(r2)}]
                          [,CHANGE={ON|OFF}]
                          [,CONTROL={E|S}]
                          [,LOCKOPT={1|2}]
                          [,KEEP={NO|YES}]
                          [,OWNER={TASK|PARTITION}]
                          [,SCOPE={INT|EXT}]
------------------------------------------------------------
```

The MODDTL macro modifies operands (fields) of a DTL (Define The Lock) control block. A DTL is used by the LOCK/UNLOCK macros to enqueue/dequeue a specific resource. The control block must have been generated by the DTL or GENDTL macro.

Operands not specified in the MODDTL macro leave the corresponding field in the DTL unchanged. There are no default values for the MODDTL macro.

ADDR={name1|(S,name1)|(r1)}: Specifies the address of the DTL.

NAME={name2|(S,name2)|(r2)}: Specifies the address of the area where a 12-byte long resource name is stored. If the name is shorter than 12 bytes, it must be padded with blanks. It is by this name, that VSE controls shared access of the resource as requested by active tasks via the LOCK macro. These tasks may all be active in one partition, or they may be distributed over several partitions; the resource-share control extends across partitions.

Note that the name must not begin with any of the characters A through I or V, because these characters are reserved for IBM usage.

CHANGE={ON|OFF}: CHANGE=ON sets up the DTL such that a subsequent UNLOCK macro would not release the resource, but reduce its locking status. Reducing the lock status can be done only when the current lock status is defined with strongest possible values: CONTROL=E and LOCKOPT=1. At least one of the operands CONTROL and LOCKOPT should be specified, too. CHANGE=OFF causes a subsequent UNLOCK macro to resume its normal function: to dequeue the resource.

CONTROL={E|S}: Defines how the named resource can be shared while your program owns it, which is determined by this specification and your specification for the operand LOCKOPT. A specification of E means the resource is enqueued for exclusive use; a specification of S means the resource is enqueued as sharable.

LOCKOPT={1|2}: This operand, together with the CONTROL parameter, determines how the system controls shared access in response to a LOCK request.

- LOCKOPT=1 and CONTROL=E: No other task is allowed to use the resource concurrently.
- LOCKOPT=1 and CONTROL=S: Other 'S' users are allowed concurrent access, but no concurrent 'E' user is allowed.
- LOCKOPT=2 and CONTROL=E: No other 'E' user gets concurrent access; however, other 'S' users can have access to the resource.
- LOCKOPT=2 and CONTROL=S: Other 'S' users can have concurrent access and, in addition, one 'E' user is allowed.

All users of a particular resource have to use the same LOCKOPT specification when they lock the resource. Exception: If LOCKOPT=1 and CONTROL=E, the lock status may be modified.

KEEP={NO|YES}: This operand may be used to lock the named resource beyond job step boundaries. Only a main task should use this operand. KEEP=NO indicates that the named resource once locked, is to be released automatically at the end of the particular job step. With KEEP=YES, a named resource that is locked remains locked across job steps; it will be automatically released at end-of-job. If a job terminates abnormally, all resources with KEEP=YES are unlocked by the abnormal termination routine.

OWNER={TASK|PARTITION}: Defines whether the named resource, once locked, can be unlocked only by the task which issued the corresponding LOCK request (OWNER=TASK), or whether it can be unlocked by any task within the partition (OWNER=PARTITION).

When OWNER is defined as PARTITION, a LOCK request for the resource must not specify FAIL=WAIT or FAIL=WAITC because deadlock prevention (return code 16) is not supported with OWNER=PARTITION.

SCOPE={INT|EXT}: This operand may be used for locking resources across systems. SCOPE=EXT specifies that the lock is used across systems. Specify SCOPE=INT if the locking is to apply to one system only.

MRMOD

```
--------------------------------------------------------
Name          Operation     Operand
--------------------------------------------------------
[name]        MRMOD         [ADDRESS={SINGLE|DUAL}]
                            [,BUFFERS=n]
                            [,SEPASMB=YES]
--------------------------------------------------------
```

The MRMOD macro defines a logic module for a MICR or OCR file. If a module name is omitted, one of the following standard module name is generated by IOCS:

IJUSZZZZ or IJUDZZZZ

S = single address adapter
D = dual address adapter

ADDRESS={SINGLE|DUAL}: Required only if the dual address adapter is used for the 1419 or 1275. If the operand is omitted, the single address adapter is assumed by the assembler.

BUFFERS=n: A numeric value equal to the corresponding value specified in the DTFMR macro.

SEPASMB=YES: Include this operand only if the module is to be assembled separately. This causes a CATALR card with the filename to be punched ahead of the object deck and the filename to be defined as an ENTRY point in the assembly. If the operand is omitted, the assembler assumes that the DTF is assembled together with the problem program.

MTMOD

```
------------------------------------------------------------
Name          Operation    Operand
------------------------------------------------------------
[name]        MTMOD        [ASCII=YES]
                           [,CKPTREC=YES]
                           [,ERREXT=YES]
                           [,ERROPT=YES]
                           [,NOTEPNT={YES|POINTS}]
                           [,RDONLY=YES]
                           [,READ={FORWARD|BACK}]
                           [,RECFORM=xxxxxx]
                           [,SEPASMB=YES]
                           [,TYPEFLE={OUTPUT|INPUT|WORK}]
                           [,WORKA=YES]
------------------------------------------------------------
```

The MTMOD macro defines a logic module for a magnetic tape file.

ASCII=YES: Include this operand if processing of ASCII input or output files is required (see Appendix B). If this operand is omitted, EBCDIC processing is assumed. ASCII=YES is not permitted for work files.

CKPTREC=YES: This operand is necessary if an input tape has checkpoint records interspersed among the data records. The module also processes tapes that do not have checkpoint records; that is, those whose DTFs do not specify CKPTREC=YES.

This operand is not needed for work files and must not be included when ASCII=YES.

ERREXT=YES: Include this operand if additional I/O errors are to be indicated and/or the ERET macro is used with this DTF and module. ERROPT=YES should be specified in this module for work files, but is not needed for input or output files.

ERROPT=YES: Include this operand if the module is to handle any of the error options for an error block. Code is generated to handle any of the three options (IGNORE, SKIP, or name). The module also processes any files in which the ERROPT operand is not specified in the DTF. This entry is needed for work files, but it is not needed for input or output files.

NOTEPNT={POINTS|YES}: This operand applies only to work files (EBCDIC only). If YES is specified, the NOTE, POINTW, POINTR, or POINTS macros can be issued for a tape work file. If POINTS is specified, only POINTS macros can be issued for tape work files.

Modules specifying either one of the two options also process work files for which the NOTE/POINTx operand is not specified in the DTF. Modules specifying YES also process work files specifying only POINTS.

RDONLY=YES: This operand causes a read-only module to be generated. Whenever this operand is specified, any DTF used with the module must have the same operand.

Each time a read-only module is entered, register 13 must contain the address of a 72-byte doubleword-aligned save area. Each task should have its own uniquely defined save area, and each time an imperative macro (except an OPEN, OPENR, or LBRET) is issued, register 13 must contain the address of the save area associated with the task. The fact that the save areas are unique for each task makes the module reentrant (that is, capable of being used concurrently by several tasks).

If the operand is omitted, the module generated is not reenterable and no save area is required.

READ={FORWARD|BACK}: This operand generates a module that reads tape files forward or backward. If FORWARD is specified, only code to read tape forward is generated. Any DTF used with the module must not specify BACK in the READ operand.

If BACK is specified, code to read tape files both forward and backward is generated, and any DTF used with the module may specify either FORWARD or BACK as its READ operand. READ=BACK does not handle multi-volume files.

This entry is not needed for work files.

RECFORM={FIXUNB|FIXBLK|VARUNB|VARBLK|SPNBLK|SPNUNB|UNDEF}: This operand generates an input/output module that processes either EBCDIC or ASCII fixed-length, variable-length, or undefined records.

If FIXUNB or FIXBLK is specified, a module is generated that allows processing of both fixed-length blocked and fixed-length unblocked records. Similarly, if VARUNB/SPNUNB or VARBLK/SPNBLK is specified, a module is generated that allows processing of both types of variable and spanned records. ASCII files are not permitted in spanned record format.

If UNDEF is specified, a module for processing undefined record types is generated. Any DTF used with the module must specify the same record format type as the module. For example, if the module specifies RECFORM=FIXUNB, either RECFORM=FIXUNB or RECFORM=FIXBLK may be specified in the DTF.

This operand is not needed for work files.

If this operand is omitted, the module generated will allow processing of both fixed-length blocked and fixed-length unblocked records.

SEPASMB=YES: Include this operand only if the module is to be assembled separately. This causes a CATALR card with the module name (standard or user-specified) to be punched ahead of the object deck and the module to be defined as an ENTRY point in the assembly. If the operand is omitted, the assembler assumes that the module is assembled together with the DTF and the problem program, and no CATALR card is punched.

TYPEFLE={OUTPUT|INPUT|WORK}: This operand generates a module that processes either GET/PUT macros or READ/WRITE, NOTE/POINTx and CHECK macros for work files (EBCDIC only). If WORK is specified, code to process work files is generated. Otherwise, a module to handle both input and output files is assumed. Only DTFs for work files may be used with work file modules. Only DTFs for input or output files may be used with an input/output module.

> **Note:** INPUT and OUTPUT have the same table format and logic modules.

WORKA=YES: If I/O records are processed in work areas instead of I/O areas, specify this operand. WORKA=YES is required for spanned record processing. The module also processes files that do not use a work area. The operand is not needed for work files.

Standard MTMOD Names

Each name begins with a 3-character prefix (IJF) and continues with a 5-character field corresponding to the options permitted in the generation of the module.

In MTMOD there are two module classes: the module class for handling GET/PUT functions and the module class for handling READ/WRITE, NOTE/POINTx, and CHECK functions (work files). Modules handling fixed-length (F and X) and undefined (U and N) records are mutually exclusive of each other and of all forms of the module that process variable-length records (V, R, and S).

Name list for GET/PUT type modules:

MTMOD name = IJFabcde

a	= F	RECFORM=FIXUNB or FIXBLK, EBCDIC mode
	= X	RECFORM=FIXUNB or FIXBLK, ASCII mode
	= V	RECFORM=VARUNB or VARBLK, EBCDIC mode
	= R	RECFORM=VARUNB or VARBLK, ASCII mode
	= S	RECFORM=SPNUNB or SPNBLK
	= U	RECFORM=UNDEF, EBCDIC mode
	= N	RECFORM=UNDEF, ASCII mode
b	= B	READ=BACK
	= Z	READ=FORWARD, or if READ is not specified
c	= C	CKPTREC=YES
	= Z	CKPTREC=YES is not specified

d = W	WORKA=YES
= Z	WORKA=YES is not specified

e = M	ERREXT=YES, RDONLY=YES
= N	ERREXT=YES
= Y	RDONLY=YES
= Z	ERREXT and RDONLY not specified

Name list for work file type modules (TYPEFLE=WORK):

MTMOD name = IJFabcde

a = W	Always

b = E	ERROPT=YES
= Z	ERROPT is not specified

c = N	NOTEPNT=YES
= S	NOTEPNT=POINTS
= Z	NOTEPNT is not specified

d = Z	Always

e = M	ERREXT=YES and RDONLY=YES
= N	ERREXT=YES
= Y	RDONLY=YES
= Z	ERREXT and RDONLY not specified

Subset/Superset MTMOD Names

The following chart shows the subsetting and supersetting allowed for MTMOD names. Four of the GET/PUT parameters allow subsetting. For example, the module IJFFBCWZ is a superset of the module IJFFBZWZ specifying fixed-length records.

```
-------------------------------------------
For GET/PUT Type Modules:
            * + + + +
    I J F F B C W M
            N Z Z Z Y
            R       +
            U       N
            X       Z
            +
            S
            V

For Workfile Type Modules:
              + +   +
    I J F W E N Z M
              Z S   Y
                Z   +
                    N
                    Z
-------------------------------------------
   + Subsetting/supersetting permitted.
   * No subsetting/supersetting permitted.
-------------------------------------------
```

MVCOM

```
------------------------------------------------------
Name        Operation    Operand
------------------------------------------------------
[name]      MVCOM        to,length,{from|(0)}
------------------------------------------------------
```

The MVCOM macro modifies the content of bytes 12 through 23 of the communication region of the partition from which the macro is issued. This area is commonly referred to as the user area.

The following example shows how to move three bytes from the symbolic location DATA into bytes 16 through 18 of the communication region:

```
MVCOM  16,3,DATA
```

to: Specifies the address (relative to the first byte of the region) of the first communication region byte to be modified.

length: Represents the number of bytes (1 to 12) to be inserted.

from|(0): Represents the address (either as a symbol or in register notation) of the bytes to be inserted.

NOTE

Name	Operation	Operand
[name]	NOTE	{filename|(1)}

The NOTE macro obtains identification for a physical record or logical block that is read or written during processing. At least one READ or WRITE operation should be successfully completed by means of the CHECK macro before issuing the NOTE macro. To NOTE a desired record successfully, the POINTR, POINTS, or POINTW macros must not be issued between CHECK and NOTE.

For magnetic tape, the last record read or written in the specified file is identified by the number of physical records read or written from the load point. The physical record number is returned in binary in the three low-order bytes of register 1. The high-order byte contains binary zero.

For CKD DASD, the binary number returned in register 1 is in the form cchr, where

cc	= cylinder number,
h	= track number,
r	= record number within the track.

Register 0 contains the unused space remaining on the track following the end of the identified record.

For FBA devices, register 1 contains an address relative to the beginning of the file in the form cccb, where ccc is the relative number of the current control interval (origin 0), and b is the relative block number within the current CI (origin 1). Register 0 contains the length of the longest logical block that could completely fit in the CI following the NOTEd logical block.

You must provide a four- or six-byte field and store in it the record identification and the remaining capacity so that it can be used later by a POINTR or POINTW macro to find the NOTEd record again. The two-byte track or CI capacity remaining is needed only when a WRITE SQ is to follow the POINTR or POINTW.

OPEN, OPENR

```
 ---------------------------------------------------
 Name        Operation    Operand
 ---------------------------------------------------
 [name]      OPEN         {filename1|(r1)}
                          [,filename2|,(r2)],...
 ---------------------------------------------------
```

The format of the OPENR macro is the same as that of the OPEN macro, except that you code OPENR instead of OPEN in the operation field.

The OPEN or OPENR macro activates all files.

When OPENR is specified, the symbolic address constants that OPENR generates from the parameter list are self-relocating. When OPEN is specified, the symbolic address constants are not self-relocating. Throughout the manual the term OPEN also implies OPENR, unless stated otherwise.

OPEN need not be issued for DTFCN and DTFPT files in a non-self-relocating environment. However, self-relocating programs using LIOCS must specify OPENR for all files, including console files.

If OPEN attempts to activate a file whose device is unassigned, the job is terminated. If the device is assigned IGN, OPEN does not activate the file, but turns on the DTF byte 16, bit 2, to indicate that the file is not activated. If DTF byte 16, bit 2 is on after issuing an OPEN, I/O operations should not be attempted for the file, as unpredictable results may occur.

filename|(r1): Enter the symbolic name of the file (DTF filename) to be opened in the operand field. A maximum of 16 files may be opened with one OPEN or OPENR by entering additional filenames. Alternatively, you can load the address of the filename in a register and specify the register using ordinary register notation.

ORMOD

```
-----------------------------------------------------------
Name          Operation    Operand
-----------------------------------------------------------
[name]        ORMOD        [BLKFAC=YES]
                           [,CONTROL=YES]
                           ,DEVICE={1287D|1287T}
                           [,IOAREA2=YES]
                           [,RECFORM={FIXUNB|FIXBLK|UNDEF}]
                           [,SEPASMB=YES]
                           [,WORKA=YES]
-----------------------------------------------------------
```

The ORMOD macro defines a logic module for a 1287 or 1288 optical reader file.

> **Note:** ORMOD is not used for the 3881 Optical Mark Reader. The 3881 uses CDMOD.

BLKFAC=YES: Include this operand if RECFORM=UNDEF and groups of undefined journal tape records are to be processed as blocks of data. (See the DTFOR BLKFAC=n operand.) The DTFOR used with this module must also include RECFORM=UNDEF and BLKFAC=n.

CONTROL=YES: Include this operand if CNTRL macros are to be used with the associated DTFs. The module also processes files that do not use the CNTRL macro.

DEVICE={1287D|1287T}: This operand must be included to specify the I/O device associated with this file. 1287D specifies a 1287 or 1288 document file. 1287T specifies a 1287 journal tape file.

IOAREA2=YES: Include this operand (journal tape only) if a second I/O area is used. The DTFOR used with this module must also include the IOAREA2 parameter.

RECFORM={FIXUNB|FIXBLK|UNDEF}: This operand generates a module that processes the specified record format. Any DTF used with the module must have the same operand.

SEPASMB=YES: Include this operand only if the module is to be assembled separately. This causes a CATALR card with the filename to be punched ahead of the object deck and the filename to be defined as an ENTRY point in the assembly. If the operand is omitted, the assembler assumes that the DTF is assembled together with the problem program.

WORKA=YES: Include this operand (journal tape only) if records are to be processed in work areas instead of in I/O areas. Any DTF used with the module must have the same operand.

ORMOD

Standard ORMOD Names

Each name begins with a 3-character prefix (IJM) followed by a 5-character field corresponding to the options permitted in the generation of the module.

ORMOD name = IJMabcde

```
a = F  RECFORM=FIXUNB
  = X  RECFORM=FIXBLK
  = U  RECFORM=UNDEF
  = D  RECFORM=UNDEF and BLKFAC=YES

b = C  CONTROL=YES
  = Z  CONTROL=YES is not specified

c = I  IOAREA2=YES
  = W  WORKA=YES
  = B  both are specified
  = Z  neither is specified

d = T  device is in tape mode
  = D  device is in document mode

e = Z  always
```

Subset/Superset ORMOD Names

The following chart shows the subsetting and supersetting allowed for ORMOD names. One of the parameters allows subsetting. For example, the module IJMFCITZ is a superset of the module IJMFZITZ.

```
----------------------------------------
             * + * *
       I J M D C B D Z
             F Z I T
             U   W
             X   Z
----------------------------------------
+ Subsetting/supersetting permitted
* No subsetting/supersetting permitted
----------------------------------------
```

PAGEIN

You can code the macro in either of the following two formats:

```
--------------------------------------------------------
Name          Operation     Operand
--------------------------------------------------------
[name]        PAGEIN        beginaddr,endaddr
                            [,beginaddr,endaddr]...
                            [,ECB={ecbname|(0)}]

[name]        PAGEIN        {listname|(1)}
                            [,ECB={ecbname|(0)}]
--------------------------------------------------------
```

The PAGEIN macro causes specific areas to be brought into real storage before their contents are needed by the requesting program. If the requested area is already in real storage the attached page frame will get low priority for the next page-outs. This function, however, does not include any fixing, so that it cannot determine whether all areas requested will still be in real storage when the entire request has been completed.

The system can handle up to 15 active PAGEIN requests at any point in time. On a system that was generated with VM=YES (in the SUPVR generation macro), execution of the macro results in a null operation. If the ECB operand is specified, the ECB will be posted.

beginaddr: Points to the first byte of the area to be paged in.

endaddr: Points to the last byte of the area to be paged in.

listname|(1): Is the name of a list of consecutive 8-byte entries as shown below.

```
┌──────────────────────────────────────────────────┐
│ X'00' | address constant      | length minus 1   │
└──────────────────────────────────────────────────┘
0       1                       4                  7
```

where:

address constant	= Address of the first byte of the area to be paged in.
length	= A binary constant indicating the length of the area to be paged in.

A non-zero byte following an entry indicates the end of the list.

ECB=ecbname(0): Specifies the name of the ECB, a fullword defined by your program, which is to be posted when the operation is complete. An invalid ECB address causes the task to be canceled.

Return Information

The return information can be obtained from the ECB, byte 2. The meaning of these bits is shown below.

Bit	**Meaning if bit is one:**
0	PAGEIN request is finished.
1	The page table is full, the request cannot be queued at this time for further handling; the request is ignored, bit 0 is set.
2	One or more of the requested pages are outside the requesting program's partition; PAGEIN is not performed for these pages.
3	At least one negative length has been detected in the area specifications; PAGEIN is not performed for these areas.
4	List of areas that are to be paged in is not completely in the requesting program's partition; the request is ignored, bit 0 is set.
5	Paging activity is too high in the system, no performance improvement is possible; the request is terminated, bit 0 is set.
6-7	Reserved.

Any combination of the return bits in the ECB is possible.

Use the WAIT macro with the ecbname as operand for completion of the PAGEIN macro, before the bits in byte 2 of the ECB are tested.

The PAGEIN function runs asynchronously with the requesting user task; therefore, if no ECB has been specified, the requesting task cannot be notified when the PAGEIN function is completed.

PDUMP

```
-----------------------------------------------------------
Name        Operation    Operand
-----------------------------------------------------------
[name]      PDUMP        {address1|(r1)},{address2|(r2)}
                         [,MFG={area|(S,area)|(r3)}]
-----------------------------------------------------------
```

This macro provides a hexadecimal dump of the general registers and of the virtual storage area contained between the two address expressions (address1 and address2). The contents of registers 0 and 1 are over-written, but the CPU status is retained. Thus, PDUMP furnishes a dynamic dump (snapshot) useful for program checkout. Processing continues with your next instruction.

The dump is always directed to SYSLST with 121-byte records. If SYSLST is not assigned, the PDUMP macro is ignored. The first byte is an ASA control character. When SYSLST is a disk drive, you must issue an OPEN macro to any DTF assigned to SYSLST after each PDUMP that is executed. The OPEN macro updates the disk address maintained in the DTF table to agree with the address where the PDUMP output ends. If the OPEN is not issued, the address is not updated, and the program is canceled when the next PUT is issued.

If non-addressable areas were included in the range of PDUMP, a message will be printed to indicate this.

{address1|(r1)},{address2|(r2)}: One or both of the addresses can be specified in register notation. If address2 is not greater than address1, or address1 is greater than the highest address in the allocated virtual storage, the macro results in no operation. If the value in address2 is greater than the end of the allocated virtual storage area, the virtual storage between address1 and the end of the allocated virtual storage is dumped.

MFG={area|(S,area)|(r3)}: The MFG operand is required if the program which issues the PDUMP is to be reenterable. It specifies the address of a 64-byte dynamic storage area, that is, storage which you obtained by a GETVIS macro; this area is needed by the system during execution of the macro.

PFIX

You can code the macro in either of the following two formats:

Name	Operation	Operand
[name]	PFIX	beginaddr,endaddr [,beginaddr,endaddr]...
[name]	PFIX	{listname|(1)}

The PFIX macro causes specific pages to be brought into real storage and fixed in their page frames until they are released at some later time. The maximum number of pages that may be fixed at any one time is specified via the ALLOCR command. Each time a page is fixed a counter for that page is incremented. This counter may never exceed 255 for any page.

If your supervisor was generated with VM=YES (in the SUPVR generation macro), execution of the macro results in a null operation; the return code is set to zero.

beginaddr: Points to the first byte of the area to be fixed.

endaddr: Points to the last byte of the area to be fixed.

listname|(1): Is the name of a list of consecutive 8-byte entries as shown below.

```
┌──────────────────────────────────────────────┐
│ X'00' | address constant    | length minus 1  │
└──────────────────────────────────────────────┘
0       1                     4                 7
```

where:

address constant	= Address of the first byte of the area to be fixed.
length	= A binary constant indicating the length of the area to be fixed.

A non-zero byte following an entry indicates the end of the list. Register notation may be used.

Exceptional Conditions

- If a PFIX causes the count of fixes for a page to exceed 255, the task issuing the PFIX is canceled.
- If it is not possible to fix all pages requested, then none will be fixed.
- If PFIX is issued in a program running in real mode, it is ignored and register 15 contains 0.

Return Codes in Register 15

0	The pages were successfully fixed.
4	The number of pages to be fixed for one request exceeds the number of PFIXable page frames; in order for this PFIX request to be satisfied, more PFIXable storage must be allocated through the ALLOCR command.
8	Not enough page frames are available in the partition because of previous PFIXes or current system resource usage; this PFIX request could, however, be satisfied at another time without reallocating PFIXable storage.
12	One of the specified addresses was invalid.

PFREE

You can code the macro in either of the following two formats:

Name	Operation	Operand
[name]	PFREE	beginaddr,endaddr [,beginaddr,endaddr]...
[name]	PFREE	{listname|(1)}

Each page in the virtual address area is assigned a 'PFIX counter'. If a page is not fixed - that is, if it is subject to normal page management - the counter is 0. Whenever a page is fixed by using a PFIX macro its counter is increased by one. All pages whose counters are greater than 0 remain fixed in real storage.

The PFREE macro decrements the counter of a specified page by 1. If a PFREE is issued for a page whose counter is 0, that PFREE is ignored since the page has already been freed.

If your supervisor was generated with VM=YES (in the SUPVR generation macro), execution of the macro results in a null operation; the return code is set to zero.

beginaddr: Points to the first byte of the area to be freed.

endaddr: Points to the last byte of the area to be freed.

listname(1): Is the symbolic name of a list of consecutive 8-byte entries as shown below.

```
┌─────────────────────────────────────────────────┐
│ X'00' | address constant    | length minus 1     │
└─────────────────────────────────────────────────┘
0        1                    4                    7
```

where:

address constant	= Address of the first byte of the area to be freed.
length	= A binary constant indicating the length of the area to be freed.

A non-zero byte following an entry indicates the end of the list.

<u>Exceptional Conditions</u>

If PFREE is issued by a program running in real mode, the macro is ignored.

<u>Return Codes in Register 15</u>

0	The pages were successfully freed.
12	One of the specified addresses was invalid.

POINTR

```
-------------------------------------------------------
Name         Operation    Operand
-------------------------------------------------------
[name]       POINTR       {filename|(1)}
                          ,{address|(0)}
-------------------------------------------------------
```

The POINTR macro repositions the file specified by filename to the record identified by previously issuing a NOTE macro.

If a READ follows the POINTR, the NOTEd record is the record read (tape or DASD).

For magnetic tape, a WRITE must not follow a POINTR.

For DASD work files, if a WRITE UPDATE follows the POINTR, the NOTEd record is written (or overwritten). If a WRITE SQ follows the POINTR, the record after the NOTEd record is written (or overwritten) and, on CKD DASD, the remainder of the track is erased (overwritten with zeros). On FBA devices, the remainder of the CI is erased (overwritten with zeros) and an SEOF is written (the following CI is also overwritten with zeros).

filename|(1): The filename may be expressed either as a symbol or in register notation.

address|(0): Specifies a virtual storage location in which is stored either a four-byte record identifier or a four-byte record identifier plus a two-byte track or CI capacity. The four- or six-byte number must be in the form obtained from the NOTE macro. The two-byte track or CI capacity is required only when a WRITE SQ is to be issued following the POINTR.

POINTS

```
-----------------------------------------------------
 Name          Operation     Operand
-----------------------------------------------------
 [name]        POINTS        {filename|(1)}
-----------------------------------------------------
```

The POINTS macro repositions a file to its beginning.

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

For disk work files, the file is repositioned to the lower limit of the first extent. A POINTS should not be followed by a WRITE UPDATE. If a POINTS is followed by a WRITE SQ, the first record in the file is overwritten. For CKD DASD, the remainder of the track is then erased (overwritten with zeros). For FBA devices, the remainder of the CI is erased (overwritten with zeros) and an SEOF is written (the following CI is also overwritten with zeros).

filename|(1): The filename may be expressed either as a symbol or in register notation.

```
-------------------------------------------------
Name          Operation     Operand
-------------------------------------------------
[name]        POINTW        {filename|(1)}
                            ,{address|(0)}
-------------------------------------------------
```

The POINTW macro repositions the file specified by filename to the record following the record identified by previously issuing a NOTE macro. A READ or WRITE following a POINTW macro results in the following:

- For magnetic tape, a READ following a POINTW causes the record following the NOTEd record to be read.
- For DASD work files, a READ following a POINTW causes the NOTEd record to be read.
- For magnetic tape, a WRITE UPDATE following a POINTW causes the record following the NOTEd record to be overwritten.
- For DASD work files, a WRITE UPDATE following a POINTW causes the NOTEd record to be overwritten.

If a WRITE SQ follows the POINTW, the record after the NOTEd record is written (or overwritten) and, on CKD DASD, the remainder of the track is erased (overwritten with zeros). On FBA devices, the remainder of the CI is erased (overwritten with zeros) and an SEOF is written (the following CI is also overwritten with zeros).

filename|(1): The filename may be expressed either as a symbol or in register notation.

address|(0): Specifies a virtual storage location in which is stored either a four-byte record identifier or a four-byte record identifier plus a two-byte track or CI capacity. The four- or six-byte number must be in the form obtained from the NOTE macro. The two-byte track or CI capacity is required only when a WRITE SQ is to be issued following the POINTW.

POST

```
-----------------------------------------------------
Name          Operation     Operand
-----------------------------------------------------
[name]        POST          {ecbname|(1)}
                            [,SAVE={savearea|(0)}]
-----------------------------------------------------
```

This macro provides intertask communication by posting an ECB (it turns on byte 2, bit 0). A POST issued to an ECB removes a task waiting for the ECB from the wait state.

ecbname|(1): Provides the address of the ECB that is to be posted.

SAVE={savearea|(0)}: This operand may be used for taking a specific waiting task out of the wait state. The operand causes the system to locate the save area whose address is specified in the operand and to take only the subtask associated with this save area out of the wait state. This task normally is waiting for the specified ECB to be posted.

Although time is saved by specifying this operand, other tasks waiting for this ECB are not taken out of the wait state for this event by this issuance of the POST macro. This does not guarantee that they will stay in the wait state until another POST is issued. On the contrary, other events could cause the other tasks to be dispatched. For this reason the POST macro should not be used with the SAVE operand to control subtask operation unless separate ECB's are used. Otherwise, it should be used only to save time. When a POST is issued without the SAVE operand, all tasks waiting for the ECB are taken out of the wait state, and the highest priority task regains control.

```
-------------------------------------------------------------
Name        Operation   Operand
-------------------------------------------------------------
[name]      PRMOD       [CONTROL=YES]
                        [,CTLCHR={YES|ASA}]
                        [,DEVICE=xxxxx]
                        [,ERROPT=YES]
                        [,FUNC=xxx]
                        [,IOAREA2=YES]
                        [,PRINTOV=YES]
                        [,RDONLY=YES]
                        [,RECFORM={FIXUNB|VARUNB|UNDEF}]
                        [,SEPASMB=YES]
                        [,STLIST=YES]
                        [,TRC=YES]
                        [,WORKA=YES]
-------------------------------------------------------------
```

The PRMOD macro defines a logic module for a printer file.

If advanced printer buffering is used on your 3800 Printer Subsystem, the PRMOD macro is not needed.

CONTROL=YES: Include this operand if CNTRL macros are used with the associated DTFs. The module also processes files that do not use the CNTRL macro. If CONTROL is specified, the CTLCHR operand must not be specified.

The CONTROL operand is not allowed for the 2560 or 5424/5425.

CTLCHR={YES|ASA}: Include this operand if first-character carriage control is used. Any DTF used with the module must have the same operand. If CTLCHR is specified, CONTROL must not be specified.

CTLCHR must not be specified for the 2560 or 5424/5425.

If CTLCHR=ASA is specified for the 3525, the + character is not allowed. For 3525 print (not associated) files, you must issue either a space 1 command or skip to channel 1 command to print on the first line of a card. For 3525 print associated files, you must issue a space 1 command to print on the first line of a card.

If, in a multitasking environment, several DTFPRs address the same device, and at least one DTF specified CTLCHR=ASA, overprinting may occur. Therefore, while a DTFPR (with CTLCHR=ASA) is doing an I/O operation, no other DTFPR should be allowed to do I/O on the same device.

DEVICE={1403|1443|2245|2560P|2560S|3203|3211|3525|3800|5203|5425P|5425S|PRT1}: This operand specifies which device is used for the file. The 'P' and 'S' included with the '2560' and '5425' parameters specify primary or secondary input hoppers; regardless of which is specified, however, the module generated will handle DTFs specifying either hopper. Specify 5425P/S for 5424P/S.

Any DTF to be used with this module must have the same operand (except as just noted concerning the 'P' and 'S' specification for the 2560 or 5424/5425).

ERROPT=YES: This operand must be specified if ERROPT=name is specified in a DTFPR that is to be used with the module. (ERROPT=name is applicable to a PRT1 printer only.) If ERROPT is not specified in the DTFPR, or if ERROPT=RETRY (3211) or ERROPT=IGNORE (3525) is specified, ERROPT=YES must be omitted.

FUNC={W[T]|RW[T]|RPW[T]|PW[T]}: This operand specifies the type of file to be processed by the 2560, 3525, or 5424/5425. Any DTF used with the module must include the same operand. W indicates print, R indicates read, P indicates punch, and T (for the 3525 only) indicates an optional 2-line printer.

RW[T], RPW[T], and PW[T] are used to specify associated files; when one of these parameters is specified for a printer file it must also be specified for the associated file(s).

If a 2-line printer is not specified for the 3525, multi-line print is assumed. T is ignored if CONTROL or CTLCHR is specified.

IOAREA2=YES: Include this operand if a second I/O area is used. Any DTF used with the module must also include the IOAREA2 operand.

PRINTOV=YES: Include this operand if PRTOV macros are used with the associated DTFs. The module also processes any files that do not use the PRTOV macro.

This operand is not allowed for the 2560 or 5424/5425.

RDONLY=YES: This operand causes a read-only module to be generated. Whenever this operand is specified, any DTF used with the module must have the same operand.

RECFORM={FIXUNB|VARUNB|UNDEF}: This operand causes a module to be generated that processes the specified record format: fixed-length, variable-length, or undefined. Any DTF used with the module must include the same operand.

SEPASMB=YES: Include this operand only if this module is to be assembled separately. This causes a CATALR card with the filename to be punched ahead of the object deck and the filename to be defined as an ENTRY point in the assembly. If the operand is omitted, the assembler assumes that the DTF is assembled together with the problem program.

STLIST=YES: Include this operand if the selective tape listing feature (1403 only) is used. If this entry is specified, the

CONTROL, CTLCHR, and PRINTOV entries are not valid, and are ignored if supplied. If this operand is specified, RECFORM must specify FIXUNB.

TRC=YES: Include this operand to specify whether the module is to test the TRC bit in the DTFPR or iqnore that bit. If TRC=YES is specified, the generated module can process output files with table reference characters and those without.

WORKA=YES: Include this operand if records are processed in work areas instead of in I/O areas. Any DTF used with the module must have the same operand.

Standard PRMOD Names

Each name begins with a 3-character prefix (IJD) followed by a 5-character field corresponding to the options permitted in the generation of the module.

PRMOD name = IJDabcde

a = F	RECFORM=FIXUNB
= V	RECFORM=VARUNB
= U	RECFORM=UNDEF

b = A	CTLCHR=ASA
= Y	CTLCHR=YES
= C	CONTROL=YES
= S	STLIST=YES
= Z	None of these is specified
= T	DEVICE=3525 with 2-line printer
= U	DEVICE=2560
= V	DEVICE=5425

c = B	ERROPT=YES and PRINTOV=YES
= P	PRINTOV=YES, DEVICE is not 3525, and ERROPT is not specified
= I	PRINTOV=YES, DEVICE=3525, and FUNC=W[T] or omitted
= F	PRINTOV=YES, DEVICE=3525, and FUNC=RW[T]
= C	PRINTOV=YES, DEVICE=3525, and FUNC=PW[T]
= D	PRINTOV=YES, DEVICE=3525, and FUNC=RPW[T]
= Z	Neither PRINTOV nor ERROPT is specified, and DEVICE is not a 3525
= O	PRINTOV=YES not specified, DEVICE=3525, and FUNC=W[T] or omitted
= R	PRINTOV=YES not specified, DEVICE=3525, and FUNC=RW[T]
= S	PRINTOV=YES not specified, DEVICE=3525, and FUNC=PW[T]
= T	PRINTOV=YES not specified, DEVICE=3525, and FUNC=RPW[T]
= E	ERROPT=YES and PRINTOV=YES is not specified
= U	FUNC=W or omitted and DEVICE=2560 or 5425
= V	FUNC=RW and DEVICE=2560 or 5425
= W	FUNC=PW and DEVICE=2560 or 5425
= X	FUNC=RPW and DEVICE=2560 or 5425

```
d = I   IOAREA2=YES
  = Z   IOAREA2=YES is not specified

e = V   RDONLY=YES and WORKA=YES
  = W   WORKA=YES
  = Y   RDONLY=YES
  = Z   Neither is specified
```

Subset/Superset PRMOD Names

Two of the operands allow subsetting. For example, the module name IJDFCPIW is a superset of the module names IJDFCZIW and IJDFZZIW. No subsetting or supersetting of PRMOD names is allowed for the 2560 or 5424/5425.

The IBM-supplied preassembled logic modules do not have TRC=YES. The system programmer can reassemble them with TRC=YES to support 3800 table reference characters. Although the code that is generated for a module assembled with TRC=YES is different from the code that is generated for a module with TRC=NO, the module name is the same. If some, but not all PRMOD logic modules are reassembled this way, it may interfere with subsetting or supersetting.

```
-------------------------------------------
         * * * * *
   I J D F A U I V
         V Y V Z W
         U S W   Y
           T X   Z
           U +
           V P
           + Z
           C +
           Z I
             O
             +
             F
             R
             +
             +
             C
             S
             +
             D
             T
             +
             B
             E
-------------------------------------------
 + Subsetting/supersetting permitted.
 * No subsetting/supersetting permitted.
-------------------------------------------
```

```
------------------------------------------------------
Name        Operation    Operand
------------------------------------------------------
[name]      PRTOV        {filename|(1)},{9|12}
                         [,routine-name|,(0)]
------------------------------------------------------
```

The PRTOV (printer overflow) macro is used with a printer file to specify the operation to be performed when a carriage overflow condition occurs. To use this macro, the PRINTOV=YES operand must be included in the DTFPR macro.

filename|(1): This operand is required. Must be the filename, written either as a symbol or in register notation.

9|12: This operand is required. Specifies the number of the carriage control channel (9 or 12) used to indicate the overflow. When an overflow condition occurs, IOCS advances the printer carriage to the first printing line on the form (channel 1), and normal printing continues.

routine-name|(0): Specify this operand only if you prefer to branch to your own routine on an overflow condition, rather than skipping directly to channel 1. It specifies the name of the routine, as a symbol or in register notation. However, the name should never be preloaded into register 1.

If you specify the third parameter, IOCS does not advance the carriage to channel 1.

Return from the overflow routine via register 14.

PTMOD

```
------------------------------------------------------------
Name          Operation     Operands
------------------------------------------------------------
[name]        PTMOD         [DEVICE={2671|1017|1018}]
                            [,RECFORM={FIXUNB|UNDEF}]
                            [,SCAN=YES]
                            [,SEPASMB=YES]
                            [,TRANS=YES]
------------------------------------------------------------
```

The PTMOD macro defines a logic module for a paper tape file. If you do not provide a name for the module, IOCS generates a standard module name.

DEVICE={2671|1017|1018}: Required only to specify an I/O device other than 2671 used by the module. Any DTF used with the module must have the same operand. 2671 is assumed if this operand is omitted.

RECFORM={FIXUNB|UNDEF}: Required only if the operand SCAN=YES is present. If records of undefined format using the SCAN option are translated, specify the UNDEF parameter. If records of fixed unblocked format are translated, the FIXUNB parameter may be specified or omitted.

SCAN=YES: Required for records containing shift characters and/or characters that are automatically deleted by IOCS.

SEPASMB=YES: Include this operand only if the module is assembled separately. This causes a CATALR card with the module name (standard or user-specified) to be punched ahead of the object deck and defines the module name as an ENTRY point in the assembly. If the operand is omitted, the assembler assumes that the module is being assembled with the DTF and the problem program and no CATALR card is punched.

TRANS=YES: Required only if records using an unshifted code are translated and if the operand SCAN=YES is not specified.

Standard PTMOD Names

Each name begins with a 3-character prefix (IJE) and continues with a 5-character field corresponding to the options permitted in the generation of the module.

PTMOD name =IJEabcde

a	= S	SCAN=YES
	= Z	SCAN=YES is not specified

b	= T	TRANS=YES (SCAN=YES is not specified)
	= Z	TRANS=YES is not specified
c	= F	RECFORM=FIXUNB, and SCAN=YES
	= U	RECFORM=UNDEF, and SCAN=YES
	= Z	SCAN=YES is not specified, and/or DEVICE=1018
d	= 1	DEVICE=1017
	= 2	DEVICE=1018
	= Z	DEVICE=2671, or if this entry is omitted
e	= Z	always

Subset/Superset PTMOD Names

The following chart shows the PTMOD names. No subsetting or supersetting is allowed.

```
------------------------------------------------
          *  *  *  *
    I J E Z  Z  Z  Z  Z
          Z  T  Z  Z
          S  Z  F  Z
          S  Z  U  Z
          Z  Z  Z  1
          Z  T  Z  1
          S  Z  F  1
          S  Z  U  1
          S  Z  Z  2
          Z  T  Z  2
------------------------------------------------
  * No subsetting/supersetting permitted.
------------------------------------------------
```

PUT

```
---------------------------------------------------------------
Name        Operation    Operand
---------------------------------------------------------------
[name]      PUT          {filename|(1)}
                         [,workname|,(0)]
                         [,STLSP={controlfield|(r1)}]
                         [,STLSK={controlfield|(r2)}]
---------------------------------------------------------------
```

PUT writes, prints, or punches logical records which are built directly in the output area or in a specified work area. PUT can be used for any sequential output file defined by a DTF macro, and for any type of record: blocked or unblocked, fixed or variable length, and undefined. It operates much the same as GET but in reverse. It is issued after a record has been built.

filename|(1): This operand must be the same as the name of the DTF macro for the file that is being built. The operand can be specified as a symbol or in either special or ordinary register notation. The high-order eight bits of the register must be zero, or unpredictable results may occur. Use register notation if your program is to be self-relocating.

workname|(0): This operand specifies a work area name or a register (in either special or ordinary register notation) containing the address of the work area. The work area address should never be preloaded into register 1. This operand is used if records are built in a work area which you define yourself (for example, using a DS instruction). If the operand is specified, all PUTs for the named file must use either a register or a workname. PUT then moves each record from the work area to the output area.

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

Whenever a PUT macro transfers an output data record from an output area (or work area) to an I/O device, the data remains in the area until it is either cleared or replaced by other data. IOCS does not clear the area. Therefore, if you plan to build another record whose data does not use every position of the output area or work area, you must clear that area before you build the record. If this is not done, the new record will contain interspersed characters from the preceding record.

STLSP={controlfield|(r1)}: This operand specifies a control byte that allows for spacing while using the selective tape listing feature on the 1403 printer. To use this feature, the operand STLST=YES must be specified in the DTFPR. Up to 8 paper tapes may be independently spaced. The control byte is set up like any other data byte in virtual storage. You can also use ordinary register notation to provide the address of the control byte. Registers 2 through 12 are available without restriction. You determine the spacing (which occurs after printing) by setting on the bits corresponding to the tapes you want to space. The correspondence between control byte bits and tapes is as follows:

Control byte bits	0	1	2	3	4	5	6	7
Tape position	8	7	6	5	4	3	2	1

The tape position 1 is the leftmost tape on the selective tape listing device.

Note: Double-width tapes must be controlled by both bits of the control field.

STLSK={controlfield|(r2)}: This operand specifies a control byte that allows for skipping while using the selective tape listing feature on the 1403 printer. To use this feature, the operand STLIST=YES must be specified in the DTFPR. Up to 8 paper tapes may be independently skipped. The control byte is set up like any other data byte in virtual storage. You can also use ordinary register notation to provide the address of the control byte. Registers 2 through 12 are available without restriction. You determine the skipping (which occurs after printing) by setting on the bits corresponding to the tapes you want to skip. The correspondence between control byte bits and tapes is shown in the figure under "STLSP=control field", above.

PUTR

```
------------------------------------------------------
Name          Operation     Operand
------------------------------------------------------
[name]        PUTR          {filename|(1)}
                            [,{workname1|(0)},
                            {workname2|(2)}]
------------------------------------------------------
```

The PUTR (PUT with reply) macro is used for the display operator console, to issue a message to the operator which requires operator action and which will not be deleted from the display screen until the operator has issued a reply.

You may also use PUTR with the 3210 or 3215 console printer-keyboard, in which case PUTR functions the same as PUT followed by GET for these devices, but provides the message non-deletion code for the display operator console. Use of PUTR for the 3210 or 3215 is therefore recommended for compatibility if your program may at some time be run on the display operator console instead of the 3210 or 3215.

Use PUTR for fixed unblocked records (messages). Issue PUTR after a record has been built.

Do not use register 2 as base register in any of the PUTR operands.

filename|(1): This operand must be the same as the name of the DTFCN for the file that is being built. The filename can be specified as a symbol or in either special or ordinary register notation. The latter is necessary to make your programs self-relocating.

workname1|(0): This operand specifies the output work area name or a register (in either special or ordinary register notation) containing the address of the output work area. The work area address should never be preloaded into registers 1 or 2. This parameter is used if records are built in a work area which you define yourself (for example, using a DS instruction). The length of the work area is defined by the BLKSIZE parameter of the DTFCN macro. If workname1 is specified, workname2 must also be specified.

workname2|(2): This operand specifies the input work area name or a register (in either special or ordinary register notation) containing the address of the input work area. The work area address should never be preloaded into registers 0 or 1. This parameter is used if records are built in a work area which you define yourself (for example, using a DS instruction). The length of the work area is defined by the INPSIZE parameter of the DTFCN macro. If workname2 is specified, workname1 must also be specified.

Name	Operation	Operand
[name]	RCB	

The RCB macro generates an 8-byte word-aligned resource control block (RCB); this block allows you to protect a user-defined resource if the ENQ macro is issued before (and the DEQ macro is issued after) each use of the resource. The format of the RCB and its use is shown below.

Bytes	Purpose of bits
0	All bits are set to 1 to indicate that the resource has been placed in a priority queue by the ENQ macro.
1-3	Reserved.
4	bit 0=1: Another task is waiting to use the resource. Bits 1-7: Reserved.
5-7	ECB address of current resource owner.

RDLNE

Name	Operation	Operand
[name]	RDLNE	{filename|(1)}

The RDLNE macro provides selective on-line correction when processing journal tapes on the 1287 optical reader. This macro reads a line in the on-line correction mode while processing in the off-line correction mode. RDLNE should be used in the COREXIT routine only, or else the line following the one in error will be read in on-line correction mode.

If the 1287 cannot read a character, IOCS first resets the input area to binary zeros and then retries the line containing the character that could not be read. If the read is unsuccessful, you are informed of this condition via your error correction routine (specified in DTFOR COREXIT). The RDLNE macro may then be issued to cause another attempt to read the line. If the character in the line still cannot be read, the character is displayed on the 1287 display scope. The operator keys in the correct character, if possible. If the operator cannot readily identify the defective character, he may enter the reject character in the error line. This condition is posted in filename+80 for your examination. Wrong-length records and incomplete read conditions are also posted in filename+80.

filename|(1): The symbolic name of the 1287 file from which the record is to be retrieved. This name is the same as that specified in the DTFOR header entry for the file.

```
-------------------------------------------------------------
Name          Operation    Operand
-------------------------------------------------------------
[name]        READ         {filename|(1)}
                           {,SQ,{area|(0)}[,length|,(r1)|,S]
                            |,ID
                            |,KEY
                            |,OR,{name|(r2)}
                            |,DR,{name|(r3)|number,number}
                            |,MR}
-------------------------------------------------------------
```

The READ macro transfers a record or part of a record from an input file to an area in virtual storage.

filename|(1): Specifies the name of the file from which the record is to be read. The name is the same as that specified in the DTF header entry.

SQ: Required for sequential files.

area|(0): The name of the input area used by a sequential file.

length|(r1)|S: Used only for sequential files of undefined format (RECFORM=UNDEF). Specifies the actual number of bytes to be read, or the register where the number is to be found. S specifies that the entire record is to be read. The length of the record is taken from the DTF filenameL field.

ID: For DAM, specifies that the reference is to be by ID (identifier in the count area of the record).

KEY: For ISAM, KEY is required. For DAM, specifies that the record reference is to be by record key (control information in the key area of the DASD record).

OR: Signifies that the file is for a 1287 or 1288 optical character reader.

name|(r2): Specifies the CCW list address to be used to read a document from the 1287 or 1288.

DR: Indicates a 3886 Optical Character Reader is the input device.

The line number to be read and the format record for the line are specified in one of three ways:

- **name** provides the symbolic address of a 2-byte hexadecimal field containing the line number in the first byte and the format record number in the second byte.

- **(r3)** provides the number of the register that contains the address of the two-byte hexadecimal field.

- **number,number** provides the decimal line number to be read (any number from 1 through 33), followed by the format record number used to read the line (0-63).

MR: Signifies that the file is for a magnetic ink character reader (MICR).

Name	Operation	Operand
[name]	REALAD	{address|(1)}

In S/370 mode, the REALAD macro returns the real address corresponding to a specified virtual address. If issued in ECPS:VSE mode, the macro returns the specified virtual address.

address|(1): Is the virtual address to be converted. It can be specified as a symbol or in register notation.

Register 0 returns the address corresponding to the specified virtual address if and only if the virtual address points to a PFIXed page, otherwise register 0 contains 0. Thus, the macro can be used to test if a page is PFIXed.

Note: The pages of a partition running in real mode are treated as if they were fixed.

RELEASE

```
-------------------------------------------------------
Name        Operation     Operand
-------------------------------------------------------
[name]      RELEASE       (SYSnnn[,SYSnnn]...)
                          [,savearea]
-------------------------------------------------------
```

RELEASE specifies the names of programmer logical units to be released. RELEASE may be used only for units used within a given partition .

SYSnnn: Specifies the programmer logical unit that is to be released. Up to 16 units may be specified in a list, which must be enclosed in parentheses.

All the units specified are checked by the assembler to assure that no system logical units are requested for release. If system logical units are specified, an MNOTE is issued and such units are ignored. Before any release is attempted, a check is made for ownership of the unit. If the requesting partition does not own the unit, or if the unit is already unassigned, the request is ignored.

savearea: Is the name of an 8-byte word-aligned area where registers 0 and 1 are saved for your program. If the operand is not provided, the contents of registers 0 and 1 are over-written.

The macro expansion includes a unit table and loads the table's address into register 0. If the savearea operand is specified, the macro expansion saves registers 0 and 1.

If there is no permanent assignment, the device is unassigned. If the device is at permanent assignment level, no action is taken on the unit.

Recommendation: You should inform the system operator via a message that the assignment was released.

RELPAG

You can code the macro in either of the following two formats:

Name	Operation	Operand
[name]	RELPAG	beginaddr,endaddr [,beginaddr,endaddr]...
[name]	RELPAG	{listname|(1)}

The RELPAG macro causes the contents of one or more storage areas to be released. If the affected areas are in real storage when the RELPAG macro is executed, their contents are not saved but are overwritten when the associated page frames are needed to satisfy pending page frame requests.

After the RELPAG macro has been executed for an area and a location in that area is referenced again during the current program execution, the related page is attached to a page frame which contains all zeros.

The storage area is released only if it contains at least one full page. You can be sure of this only if the specified area is 4K minus 1 byte, or bigger (see Figure 22 on page 150).

beginaddr: Points to the first byte of the area to be released.

endaddr: Points to the last byte of the area to be released.

listname|(1): Is the symbolic name of a list of consecutive 8-byte entries as shown below.

```
┌──────────────────────────────────────────────────┐
| X'00' | address constant      |  length minus 1  |
└──────────────────────────────────────────────────┘
0        1                     4                  7
```

where:

address constant	= Address of the first byte of the area to be released.
length	= A binary constant indicating the length of the area to be released.

A non-zero byte following an entry indicates the end of the list.

Register notation may be used.

Exceptional Conditions

- The program is running in real mode.
- The area is, fully or partially, outside of the virtual partition of the requesting program.
- A page handling request is pending for the referenced page(s).
- The page(s) is (are) fixed. For these pages, the RELPAG request will be ignored.

Return Codes in Register 15

0 All referenced pages have been released or the request has been ignored because the requesting program is running in real mode.

2 The begin address is greater than the end address, or a negative length has been found.

4 The area, fully or partially, does not belong to the partition where the issuing program is running. The release request has only been executed for those pages which belong to the partition of the issuing program.

8 a. At least one of the requested pages is temporarily fixed (via CCW-translation) and/or PFIXed. The release request has only been executed for the unfixed pages.
b. A page handling request (page fault, temporary fix, PFIX) for at least one of the requested pages is pending (caused by asynchronous processing within a partition). The release request has not been executed for those pages which are involved in a page handling request.

16 List of areas that are to be released is not completely in the requesting program's partition. The request is ignored.

Any combination of the return codes 2, 4, and 8 is possible.

RELSE

Name	Operation	Operand
[name]	RELSE	{filename|(1)}

The RELSE (release) macro is used with blocked input records read from a DASD device, or updated on a DASD device. This macro is also used with blocked input records read from magnetic tape.

The macro allows you to skip the remaining records in a block and continue processing with the first record of the next block when the next GET macro is issued. When used with blocked spanned records, RELSE makes the next GET skip to the first segment of the next record.

filename|(1): The symbolic name of the file, specified in the DTF header entry. It is the only parameter required for this macro and can be specified as a symbol or in register notation.

RESCN

```
-------------------------------------------------
Name          Operation     Operand
-------------------------------------------------
[name]        RESCN         {filename|(1)}
                            ,(r1),(r2)[,n1][,n2]
-------------------------------------------------
```

The RESCN macro selectively rereads a field on a document if one or more defective characters make this type of operation necessary. The field is always right-justified into the area (normally within IOAREA1) that was originally intended for this field as specified in the CCW. The macro first resets this area to binary zeros.

> **Note:** For the 1287 models 3 and 4 and the 1288, this macro can only be used with READ BACKWARD commands. If used with READ FORWARD commands, the input area is not cleared. When 1288 unformatted fields are read, the RESCN macro should not be used.

filename|(1): Specifies the symbolic name of the 1287D file as specified in the DTFOR header entry for the file.

(r1): Specifies a general-purpose register from 2 to 12 into which the program places the address of the load format CCW.

(r2): Specifies a general-purpose register from 2 to 12 into which the program places the address of the load format CCW for reading the reference mark.

The previous three parameters are always required, and result in one attempted reread for the field.

n1: Allows you to specify the number of attempts (one to nine allowed) to reread the unreadable field. If this parameter is omitted, one is assumed. If n1 is omitted but n2 is specified, a comma must be coded instead of n1 to indicate its absence.

n2: Indicates one more reread which forces on-line correction of any unreadable character(s) by individually projecting the unreadable character(s) on the 1287 display scope.

The operator must key in a correction (or reject) character(s). This operand cannot be used for 1288 processing.

RETURN

Name	Operation	Operand
[name]	RETURN	(r1[,r2])

The RETURN macro restores the registers whose contents were saved and returns control to the calling program.

The operands r1,r2 specify the range of the registers to be reloaded from the save area of the program that receives control. The operands are written as self-defining values. When inserted in an LM machine instruction, the operands cause the desired registers in the range from 14 through 12 (14, 15, 0 through 12) to be restored from words 4 through 18 of the save area. If r2 is omitted, only the register specified by r1 is restored. To access this save area, register 13 must contain the save area address. Therefore, the address of the save area should be loaded into register 13 before execution of the RETURN macro.

RUNMODE

Name	Operation	Operand
[name]	RUNMODE	

The RUNMODE macro returns the following information to the program issuing it:

- Register 1 contains 0 if the issuing program is running in virtual mode.
- Register 1 contains 4 if the issuing program is running in real mode.

No operand is required for this macro.

SAVE

Name	Operation	Operand
[name]	SAVE	(r1[,r2])

The SAVE macro stores the contents of specified registers in the save area provided by the calling program.

The operands r1,r2 specify the range of the registers to be stored in the save area of the calling program. The address of this area is passed to the program in register 13. The operands are written as self-defining values so that they cause desired registers in the range of 14 through 12 (14, 15, 0 through 12) to be stored when inserted in an STM assembler instruction.

Registers 14 and 15, if specified, are saved in words 4 and 5 of the save area. Registers 0 through 12 are saved in words 6 through 18 of the save area. The contents of a given register are always stored in a particular word in the save area. For example, register 3 is always saved in word 9 even if register 2 is not saved.

If r2 is omitted, only the register specified by r1 is saved.

SECTVAL

```
--------------------------------------------------
Name        Operation    Operand
--------------------------------------------------
[name]      SECTVAL      [DDKR={name1|(0)}]
                         [,DVCTYP=name2]
--------------------------------------------------
```

The SECTVAL macro calculates the sector value of the address of the requested record on the track of a disk storage device when RPS is used. The macro returns this value in register 0.

The sector value is calculated from data length, key length, and record number information. Values are calculated for fixed or variable length and for keyed and non-keyed records.

DDKR={name1|(0)}: The information needed to calculate the sector value should be specified in the 4-byte field at name1, or in the specified register. If no operand is specified, register 0 is the default and should contain the necessary information. The four bytes of information have the format DDKR, where

DD= 2-byte field which specifies:

- for fixed length records, the data length of each record, or
- for variable length records, the number of data bytes used on the track, excluding standard R0 length up to the current record. Bit 0 of the first byte must be set on and the record field (R) or key field (K) must be non-zero.

  If the K and R fields are zero, the DD field is considered as a 16-bit integer number specifying the total number of bytes used so far on the track (except R0, but including all other overhead).

K= a 1-byte field indicating the key length:

- for fixed length records, the actual key length must be specified;
- for variable length records, any non-zero value is sufficient to indicate the presence of keys.

Note: For non-keyed records the value should be 0.

R= a 1-byte record number field which specifies the number of the record of which the sector value is being requested.

DVCTYP=name2: The device type code is specified at name2. If no operand is specified, it is assumed that byte 0 of register 1 contains the code. The following device type codes (which are the same as the device type codes generated in the DTF) are valid for:

3330, Models 1 and 2: X'04'
3330, Model 11: X'05'
3340: X'08', X'09', or X'0A'
3350: X'07'
3375: X'0B'

The calculated sector value is returned in register 0. If any errors are detected in calculating the sector value, a no-operation sector value (X'FF') is returned.

SEOV

Name	Operation	Operand
[name]	SEOV	filename

The SEOV (system end-of-volume) macro must only be used with physical IOCS to automatically switch volumes if SYSLST or SYSPCH are assigned to a tape output file. SEOV writes a tapemark, rewinds and unloads the tape, and checks for an alternate tape. If none is found, a message is issued to the operator who can mount a new tape on the same drive and continue. If an alternate unit is assigned, the macro fetches the alternate switching routine to promote the alternate unit, opens the new tape, and makes it ready for processing. When using this macro, you must check for the end-of-volume condition in the CCB.

SETDEV

```
---------------------------------------------------
Name          Operation      Operand
---------------------------------------------------
[name]        SETDEV         {filename|(1)}
                             ,{phasename|(r)}
---------------------------------------------------
```

The SETDEV macro changes format records during execution of the program. When the new format record has been loaded into the 3886, control returns to the next sequential instruction in your program. If the operation is not successful, the completion code is posted at EXITIND and control is passed to the COREXIT routine, or the job is canceled. If you issue the SETDEV macro and no documents remain to be processed and the end-of-file key has been pressed on the device, control is passed to the end-of-file routine.

filename|(1): Specifies the same name as that used in the DTFDR header entry. Register notation must be used if your program is to be self-relocating.

phasename|(r): Specifies the name of the format record to be loaded; or indicates the register containing the address of an 8-byte area that contains the phasename.

SETFL

Name	Operation	Operand
[name]	SETFL	{filename|(0)}

The SETFL (set file load mode) macro causes ISAM to set up the file so that the load or extension function can be performed. This macro must be issued whenever the file is loaded or extended.

When loading a file, SETFL preformats the last track of each track index. When extending a file, SETFL preformats only the last track of the last track index plus each new track index for the extension of the file. This allows prime data on a shared track to be referenced even though no track indexes exist on the shared track.

filename|(0): The name of the file loaded is the only parameter required for this macro and is the same as that specified in the DTFIS header entry for the file. It can be specified as a symbol or in register notation. Register notation is necessary if your program is to be self-relocating.

SETIME

```
--------------------------------------------------------
Name         Operation    Operand
--------------------------------------------------------
[name]       SETIME       {timervalue|(1)}
                          [,tecbname|,(r)][,PREC]
--------------------------------------------------------
```

The SETIME macro sets the interval timer to the specified value. If the tecbname operand is specified, bit 0 of byte 2 in the TECB is set to 0 so that a subsequent WAIT/WAITM macro can be issued by the task issuing the SETIME macro. A SETIME macro **without** the tecbname operand is used in combination with a previous STXIT IT macro.

> **Note:** Any previous STXIT IT specification is overwritten when using the SETIME macro **with** the tecbname operand.

When the interval specified in timervalue elapses, the interrupt routine is entered or the TECB is posted. If tecbname is omitted, the interrupt routine specified in a previous STXIT IT macro is entered (if no STXIT IT macro was issued prior to the time of the interrupt, the interrupt is ignored). If tecbname is specified, the TECB is posted (bit 0 of the TECB is set to 1) and the task is posted ready to run if it is already waiting.

timervalue|(1): Specifies the amount of time for the interval. This value can be specified either as an absolute expression or in register notation. If register notation is used, the pertinent register must contain the time value.

The largest allowable value is 55,924 seconds (equivalent to 15 hours, 32 minutes, 4 seconds) if PREC is omitted, and 8,388,607 (equivalent to 7 hours, 46 minutes, 2 seconds) if PREC is specified.

tecbname|(r): Specifies the name (address if register notation is used) of a timer event control block (TECB) which must have been defined previously in your program by a TECB macro. If you use register notation, register 0 and 1 must not be used. After having executed the SETIME macro, the system returns the TECB address in register 1.

If you omit tecbname but want to specify PREC, you must code a comma instead of tecbname to indicate the omission.

PREC: Indicates that the timer value specified in the first operand is expressed in 1/300 of a second. When PREC is omitted, the timer value is in seconds.

SETL

```
----------------------------------------------------
Name         Operation    Operand
----------------------------------------------------
[name]       SETL         {filename|(r1)}
                          ,{id-name|(r2)|KEY|BOF|GKEY}
----------------------------------------------------
```

The SETL (set limits) macro initiates the mode for sequential retrieval and initializes ISAM to begin retrieval at the specified starting address.

> **Note:** Sequential processing must always be terminated by issuing an ESETL macro. The ESETL (end set limit) macro should be issued before issuing a READ or WRITE if ADDRTR and/or RANSEQ are specified in the same DTF. Another SETL can be issued to restart sequential retrieval.

filename|(r1): Specifies the same name as that used in the DTFIS header entry, as a symbol or in register notation. Register notation is necessary if your program is to be self-relocating.

id-name|(r2): Specifies that processing is by record ID. The operand specifies the symbolic name of the 8-byte field in which you supply the starting (or lowest) reference for ISAM use. This field contains the information shown in Figure 28 on page 246 .

KEY: Specifies that processing begins with a key you supply. The key is supplied in the field specified by the DTFIS KEYARG operand. If the specified key is not present in the file, an indication is given at filenameC.

BOF: Specifies that retrieval is to start at the beginning of the logical file.

GKEY: Indicates that selected groups of records within a file containing identical characters or data in the first locations of each key can be selected. GKEY allows processing to begin at the first record (or key) within the desired group. You must supply a key that identifies the significant (high order) bytes of the required group of keys. The remainder (or insignificant) bytes of the key must be padded with blanks, binary zeros, or bytes lower in collating sequence than any of the insignificant bytes in the first key of the group to be processed. For example, a GKEY specification of D6420000 would permit processing to begin at the first record (or key) containing D642xxxx, regardless of the characters represented by the x's. Your program must determine when the generic group is completed. Otherwise, ISAM continues through the remainder of the file.

> **Note:** If the search key is greater than the highest key on the file, the filename status byte is set to X'10' (no record found).

Byte	Identifier	Contents in Hexadecimal	Information
0	m	02-F5	Number of the extent in which the starting record is located
1-2		0000 (disk)	Always zero for disk

3-4	cc	0000-00C7 (2311,2314,2319)
		0000-0193 (3330, 3333)
		0000-015B (3348 model 35)
		0000-02B7 (3348 model 70)
			Cylinder number for disk:
			for 2311, 2314, 2319: 0-199
			for 3330, 3333: 0-403
			for 3340 with 3348 model 35: 0-347
			for 3340 with 3348 model 70: 0-695

5-6	hh	0000-0009 (2311) 0000-0013 (2314, 2319) 0000-0012 (3330, 3333) 0000-000B (3340)	Head position for disk
7	r	01-FF	Record location

Figure 28. Field Supplied for SETL Processing by Record ID

SETPFA

Name	Operation	Operand
[name]	SETPFA	{entryaddress|(0)}

The SETPFA macro either establishes or terminates linkage to a page fault appendage routine that is to be entered each time a page fault occurs or is completed.

If your supervisor was generated with VM=YES (in the SUPVR generation macro), execution of the macro results in a null operation.

entryaddress|(0): If an entry address is specified, execution of the macro establishes linkage to the appendage routine. The routine at that address will be entered every time a page fault in the associated task occurs or is satisfied. The routine to be entered and all areas referenced by the routine must be fixed in real storage using the PFIX macro before SETPFA is issued. The entry address may be specified as a symbol or in register notation.

If SETPFA is issued without an operand, the linkage to the page fault appendage is terminated. Each issuance of SETPFA supersedes all previous SETPFA's for that task. Only one task per partition is allowed to have a page-fault appendage.

A page fault appendage is called only when a page fault occurs in the task owning the appendage. If a page fault occurs while a supervisor service routine is working for the owning task, the appendage is not called. The same may apply to an IBM-supplied component such as AFC/VTAM.

SETT

Name	Operation	Operand
[name]	SETT	{timervalue|(1)}

The SETT macro sets the task timer to the value, in milliseconds, specified in the operand. The largest allowable value is 21474836 milliseconds. A register can be specified, and if it is, that register must contain the number of milliseconds in binary. You can use the SETT macro only if your supervisor was generated with TTIME=partition-ID specified in the FOPT generation macro.

The SETT macro can be issued only by the main task of the partition owning the task timer. If it is issued by a program running in a partition **not** owning the task timer, the program is canceled and an error message indicating illegal SVC is printed.

SETT must not be used within an abnormal termination exit routine.

The time interval is decremented only while the task is executing. When the specified time interval has elapsed, the task timer routine supplied in the STXIT TT macro is entered.

If a routine is not supplied to the supervisor by the time of the interruption, the interrupt is ignored. When a program is restarted from a checkpoint, timer intervals set by a SETT macro are not restarted.

STXIT

```
--------------------------------------------------------
Name         Operation    Operand
--------------------------------------------------------
To establish linkage:
[name]       STXIT        {AB|IT|OC|PC|TT}
                          ,{rtnaddr|(0)},{savearea|(1)}
                          [,OPTION={DUMP|NODUMP|EARLY}]
                          [,MFG={area|(S,area)|(r)}]

To terminate linkage:
[name]       STXIT        {AB|IT|OC|PC|TT}
--------------------------------------------------------
```

The STXIT (set exit) macro establishes or terminates linkage from the supervisor to an exit routine of your program for handling the specified condition. Linkage must be established before an interrupt occurs. Always use the EXIT macro to return from these routines.

When restarting a program from a checkpoint, any STXIT linkages established prior to the checkpoint are destroyed.

If, in an exit routine, you are issuing I/O request(s) requiring the same logic module as your main routine, you must generate a read-only module by specifying RDONLY=YES in the DTF and in the logic module. Both the main routine and the exit routine require a save area of their own. Detailed information on the save area and interrupt status is given in VSE/Advanced Functions Diagnosis: Service Aids, SC33-6099.

Hexadecimal representation	Specific abnormal termination code meaning
00	Default value for all cases other than those listed below.
0A	Access control processing error.
0B	Access violation.
0C	Operator/ICCF system request.
0D	Program check in subsystem or appendage.
0E	Page fault in subsystem or appendage.
0F	Invalid FBA DASD address for SYSFIL.
10	Normal EOJ.
11	No channel program translation for unsupported device.
12	Insufficient buffer space for channel program translation.
13	CCW with count greater than 32K.
14	Page pool too small.
15	Page fault in disabled program (not a supervisor routine).
16	Page fault in MICR stacker select or page fault appendage routine.
17	Main task issued a CANCEL macro with subtask still attached.
18	Main task issued a DUMP macro with subtask still attached.
19	Operator replied cancel as the result of an I/O error message.

Figure 29 (Part 1 of 3). Abnormal Termination Codes

Hexadecimal representation	Specific abnormal termination code meaning
1A	An I/O error has occurred (see interrupt status information).
1B	Channel failure.
1C	CANCEL ALL macro issued in another task.
1D	Main task terminated with subtask still attached.
1E	I/O error on external lock file.
1F	CPU failure.
20	A program check occurred.
21	An invalid SVC was issued by the problem program or macro.
22	Phase not found.
23	CANCEL macro issued.
24	Canceled due to an operator request.
25	Invalid virtual storage address given (outside partition).
26	SYSxxx not assigned (unassigned LUB code).

Figure 29 (Part 2 of 3). Abnormal Termination Codes

Hexadecimal representation	Specific abnormal termination code meaning
27	Undefined logical unit.
28	Reserved.
29	Reserved.
2A	I/O error on page data set.
2B	I/O error during fetch from library.
2C	Page fault appendage routine passed illegal parameter to supervisor.
2D	Program cannot be executed/restarted due to a failing storage block.
2E	Invalid resource request (possible deadlock).
2F	More than 255 PFIX requests for one page.
30	Read past a /& statement.
31	Reserved.
32	Invalid DASD address.
33	No long seek on a DASD.
35	Job control open failure.
36	Page fault in I/O appendage routine.
38	Wrong privately translated CCW.
39	Error in SYSLOG channel program.
40	ACF/VTAM error, invalid condition.
41	ACF/VTAM error, invalid condition.
42	Invalid extent information violates DASD file protection.
FF	Unrecognized cancel code.

Figure 29 (Part 3 of 3). Abnormal Termination Codes

AB: An abnormal task termination routine is entered if a task is terminated for some reason other than a CANCEL, DETACH, DUMP, JDUMP, RETURN, or EOJ macro being issued by the user program itself. Upon entry to the task's abnormal termination routine:

- Termination messages and a partition dump are produced, depending on selected options (see the OPTION operand below).
- Register 0 contains the abnormal termination code in its low order byte (see Figure 29 on page 250).
- Register 1 points to the task's abnormal-termination save area, which contains the interrupt status information and the contents of registers 0 through 15 at the time of abnormal termination.

The abnormal termination routine can then examine this data and take whatever action is necessary.

Macros which should be used in this routine are, for instance, POST and CLOSE. Macros which should **not** be used are CHKPT, ENQ, LOCK, some I/O macros, or WAIT and WAITM in combination with ECBs, since the usage of these macros may result in an abnormal termination condition or a wait condition.

> **Note:** An abnormal termination condition within the abnormal termination exit routine causes this routine to be terminated immediately. A deadlock situation may occur if a wait condition occurs within a subtask's abnormal termination exit routine that has to be posted by the main task.

After the appropriate action is taken, your abnormal termination routine may either resume processing using the EXIT AB macro (main task only) or terminate the task with CANCEL, DETACH, DUMP, JDUMP, EOJ, or RETURN (if RETURN=YES in the ATTACH macro). For a main task, the whole job is terminated if OPTION=DUMP has been specified explicitly or by default. Only the current job step is terminated if OPTION=NODUMP and the termination macro used was either DUMP or EOJ.

If your routine issues the DUMP or JDUMP macro, the system produces a storage map of the partition even if job control option NODUMP was specified. For the partition, SYSLST may be assigned to a 3211 printer. If, in addition, indexing was used before your abnormal termination routine received control, a certain number of characters on every line of the printed dump may be lost, unless you reload the printer's FCB (forms control buffer) by issuing an LFCB macro before you issue the DUMP macro. The FCB image to be loaded in this case must not have an indexing byte.

Any task in a partition can attach a subtask with an ABSAVE operand. This assumes the subtask will use the attaching task's abnormal termination routine. However, the subtask may override this specification by issuing its own STXIT AB macro.

If an abnormal termination condition occurs in a main task and linkage has not been established to an abnormal termination routine, processing in the partition is abnormally terminated. However, if the abnormal termination condition occurs in a subtask without exit linkage, only the subtask is terminated.

When subtasks are detached or canceled, associated time intervals and exit linkages are cleared.

IT: An interval timer interruption routine is entered when the specified interval elapses. If the program issuing the STXIT macro instruction is a ACF/VTAM application program, the interruption exit will not be taken while ACF/VTAM is processing any request on behalf of the application program. The exit will be taken when ACF/VTAM has completed the program's request.

An interval timer interruption is ignored if no exit linkage has been established.

If an interval timer interrupt occurs while an interval timer exit routine is still processing, the handling of the interrupt is delayed. When processing ends with EXIT IT, the IT exit routine is entered again to process the new IT interrupt. (This can only occur if a short time interval was issued in your exit routine).

OC: An operator communication interruption routine is entered when the operator presses the request key on the console and types the MSG command. In case of multitasking, only the main task can process this condition.

An operator communication interruption is ignored if no exit linkage has been established.

PC: A program check interruption routine is entered when a program check occurs. If a program check occurs in a routine being executed from the logical transient area, the job containing the routine is abnormally terminated.

A program check interruption routine can be shared by more than one task within a partition. To accomplish this, issue the STXIT macro in each subtask with the same routine address but with separate save areas. To successfully share the same PC routine, the routine must be reenterable, that is, it must be capable of being used concurrently by two or more tasks. (The specified exit is not taken if the program check occurs while ACF/VTAM is processing a request issued by the program.)

If a program check condition occurs in a main task without exit linkage, processing in the partition is terminated. However, if this same condition occurs in a subtask, only the subtask is terminated.

TT: Linkage to a task timer interruption routine can be established only if TTIME=partition-ID was specified in the FOPT macro for supervisor assembly.

A task timer interruption routine is entered when the time interval specified in the SETT macro has elapsed. The STXIT (and EXIT) TT macro can be issued only by the main task of the partition owning the task timer. If it is issued by a program running in a partition **not** owning task timer, the program is canceled and an error message indicating illegal SVC is printed.

A task timer interruption is ignored if no exit linkage has been established. A task timer interrupt is ignored if it occurs while a task timer exit routine is still processing. (This can happen only if a short time interval was issued in your exit routine).

rtnaddr: Entry point address of the routine that processes the condition described in the first operand. Your exit routine may be located anywhere in the program.

savearea: Address of a 72-byte area in which the supervisor stores the old interrupt status information and general registers 0 through 15, in that order. Your program must have a separate save area for each routine that is included.

OPTION={DUMP|NODUMP|EARLY}: This operand can be used only when setting up linkage (STXIT AB) to an abnormal termination exit routine. It determines whether termination messages and a dump will be issued upon entry to the routine.

- If the OPTION operand is omitted or if OPTION=DUMP is specified, termination messages are issued upon entry to the abnormal termination routine. In addition, a partition dump is produced unless the job control option NODUMP is active.

- If OPTION=NODUMP is specified, neither a termination message nor a dump is produced. However, if the abnormal termination routine terminates abnormally, termination messages and the dump are given regardless of the OPTION specification in the STXIT macro.

  If your routine ends with a DUMP macro and the STXIT macro was specified without OPTION=NODUMP, you will obtain two dumps.

- If OPTION=EARLY is specified (for subsystems only), the AB exit routine will be invoked for any type of termination (normal or abnormal) and, for a main task, before propagating the termination to its subtasks.

  An exit with OPTION=EARLY can be set up only once during the whole lifetime of a task. Any subsequent request to modify or reset this exit is ignored and one of the following return codes is set in register 15:

X'00'	Exit successfully set
X'04'	Exit already set
X'08'	Reset not allowed
X'0C'	No subsystem request

This protects the early exit from being overwritten by any user code that is executed under the same task as the subsystem.

OPTION=EARLY can be set only in a subtask or in the main task and cannot be transferred via the ATTACH (ABSAVE=) macro.

MFG={area|(S,area)|(r)}: This operand is required if the AB or PC exit routines are to be reenterable. It specifies the address of a 64-byte dynamic storage area which the system needs during execution of the macro. Registers 0 and 1 may not be used for register notation.

Figure 30 shows what happens when one of the five exit conditions occurs while an STXIT routine is being processed within a particular partition.

Routine being Processed	Condition Occurring				
	AB	IT	OC	PC	TT
AB	C	D	I	D	D
IT	S	E	H	H	H
OC	S	H	I	H	H
PC	S	H	H	T	H
TT	S	H	H	H	I

Figure 30. Effect of an AB, IT, OC, PC, or TT Interrupt During STXIT Routine Execution

C Job canceled immediately without entering AB routine again.

D Interrupt is delayed and the TT or IT exit routine is entered after the EXIT AB macro is issued. If no EXIT AB is issued, the interrupt is ignored.

E Handling of new timer interrupt delayed until execution of EXIT IT for original interrupt.

H	Condition honored. When processing of new routine completes, control returns to interrupted routine.
I	Condition ignored.
S	Execution of the routine being processed is suspended, and control transfers to the AB routine.
T	Job abnormally terminated. If AB routine is present, its exit is taken. Otherwise, a system abnormal termination occurs.

Notes:

1. If an operator communication interruption routine or a program check interruption routine is in process when a timer interrupt occurs, your timer routine will be processed; when it completes, control returns to the interrupted routine.

2. If a task is using a logical transient routine when a timer interrupt occurs, your timer routine is not entered until the logical transient routine is released.

SUBSID

```
-------------------------------------------------------------
Name        Operation    Operand
-------------------------------------------------------------
[name]      SUBSID       INQUIRY
                         ,NAME={name1|(S,name1)|(r1)}
                         ,AREA={name2|(S,name2)|(r2)}
                         ,LEN={length|(r3)}
                         [,LVLTEST={NO|YES}]
                         [,MFG={name4|(r4)}]
-------------------------------------------------------------
```

The SUBSID macro allows you to make inquiries about the supervisor. The information about the supervisor (such as version number, modification number, or some indicators) is described by a byte string, which may be interpreted with the help of the macro MAPSSID.

NAME={name1|(S,name1)|(r1)}: Specifies the address of a 4-byte field containing the name SUPb, where b indicates a blank.

AREA={name2|(S,name2)|(r2)}: Specifies the address of the area into which the requested information is to be stored.

LEN={length|(r3)}: Specifies the length of the area as an integer, a self-defining term, or as a value in a register. The length to be specified can be obtained from the DSECT generated by the MAPSSID macro.

LVLTEST={NO|YES}: Specify LVLTEST=YES if it is possible that the program might make the inquiry on a prior-release supervisor that does not support the SUBSID function. This will prevent the program from being canceled. If LVLTEST=NO is specified and the inquiry is made by a program running on a supervisor without the SUBSID function, the inquiring program is canceled.

MFG={name4|(r4)}: The MFG operand is required if the program is to be reenterable. It specifies the address of a 64-byte dynamic storage area, that is: storage which your program obtained through a GETVIS macro. This area is required for system use during execution of the macro.

Return Codes in Register 15

0	Information returned.
8	Returned information truncated, because the area specified is too short. Register 0 contains the total length in the 2 rightmost bytes.
16	Name not found.
20	SUBSID function not available because this is a back-level supervisor (only if LVLTEST=YES).

TECB

```
---------------------------------------------------
Name          Operation    Operand
---------------------------------------------------
[name]        TECB
---------------------------------------------------
```

The TECB macro generates a timer event control block which can be referred to by the symbol you specify in the name field. This block contains an event bit that indicates when the time interval specified in SETIME has elapsed. The format of this block is as follows:

Byte	Purpose of bits
0-1	Reserved
2	Bit 0: If 0, time specified in SETIME has not elapsed. If 1, time specified in SETIME has elapsed. Bit 1-7: Reserved
3	Reserved

Name	Operation	Operand
[name]	TESTT	[CANCEL]

The TESTT macro can be used only if TTIME=partition ID was specified in the FOPT generation macro for supervisor assembly.

The TESTT macro is used to test the amount of time that has elapsed from a task timer interval set by an associated SETT. The TESTT macro returns the time remaining in the interval, expressed in hundredths of milliseconds in binary, in register 0.

The TESTT macro can be issued only by the main task of the partition owning the task timer. If it is issued by a program running in a partition not owning the task timer, the program is canceled and an error message indicating illegal SVC is printed.

CANCEL: If CANCEL is specified, the remaining time of the interval is canceled, and the task timer exit routine is not entered.

If the macro is issued without an operand, the macro must not contain a comment unless the comment begins with a comma.

TPIN

```
-----------------------------------------------------
Name         Operation    Operand
-----------------------------------------------------
[name]       TPIN
-----------------------------------------------------
```

The TPIN macro is available primarily for the telecommunication applications that require immediate system response. The macro causes one or more partitions (other than the one issuing the macro) to be deactivated. The number of partitions that can be deactivated is specified in the TPBAL command. The partitions to be deactivated are the ones with the lowest priorities. This request is ignored in each of the following cases:

- The operator has not made TP balancing active by means of the TPBAL command.
- None of the partitions specified in the TPBAL command contains a program running in virtual mode.
- The only partition that could be affected by TP balancing is the partition that issued the TPIN request.
- There is no paging in the system.

The TPIN macro must always be used in conjunction with the TPOUT macro. The operand field is ignored.

If your supervisor was generated with VM=YES (in the SUPVR generation macro), execution of the macro results in a null operation.

TPOUT

Name	Operation	Operand
[name]	TPOUT	

The TPOUT macro causes the system to reactivate partitions that had been deactivated by the TPIN macro.

Failure to issue the TPOUT macro can cause considerable and unnecessary performance degradation in the batch partition(s). The operand field is ignored.

If your supervisor was generated with VM=YES (in the SUPVR generation macro), execution of the macro results in a null operation.

TRUNC

```
----------------------------------------------------
Name          Operation    Operand
----------------------------------------------------
[name]        TRUNC        {filename|(1)}
----------------------------------------------------
```

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

Note that TRUNC will not necessarily cause a physical write to an FBA DASD unless PWRITE is also specified.

filename|(1): The symbolic name of the file, specified in the DTF header entry, is the only parameter required in this macro.

TTIMER

Name	Operation	Operand
[name]	TTIMER	[CANCEL]

The TTIMER macro is used to test how much time has elapsed of an interval which was set in the same task by the associated SETIME macro. The TTIMER macro returns the time remaining of the interval, expressed in hundredths of seconds in binary, in register 0.

CANCEL: If CANCEL is specified, the time interval set in that task is canceled. As a result of the TTIMER CANCEL macro, the interval timer interruption routine of the task (to which linkage may have been established by an STXIT IT macro) does not receive control. If the associated SETIME macro specified the same name of a TECB, that TECB's event bit is set on.

If the macro is issued without an operand, the macro must not contain a comment unless the comment begins with a comma.

UNLOCK

```
-----------------------------------------------------
Name        Operation    Operand
-----------------------------------------------------
[name]      UNLOCK       [{name|(S,name)|(r)}|ALL]
-----------------------------------------------------
```

The UNLOCK macro dequeues the issuing task (or partition) from the named resource, to which the task had previously been enqueued via the LOCK macro. The resource must have been defined to the system by a DTL or GENDTL macro.

In addition, the UNLOCK macro can be used to lower the lock control level. Reduction of the lock control level may be done only if the issuing task is currently locked, onto the resource, with the most stringent control level possible: CONTROL=E and LOCKOPT=1 (the CONTROL and LOCKOPT parameters are described with the DTL macro). The resource then continues to be held by the task; however, another task waiting for this resource can be dispatched again and may or may not gain shared access (see also the description of the LOCK macro). In order to use the UNLOCK macro for this purpose, the MODDTL macro must have been issued with the CHANGE operand set ON.

{name|(S,name)|(r)}: Specifies the DTL address.

ALL: Frees all resources which are locked by the task and whose DTLs were defined with KEEP=NO. If UNLOCK ALL is issued by the main task, not only the resources locked by that task are unlocked, but also those which have been locked by subtasks, with OWNER=PARTITION specified for DTL generation.

Return Codes in Register 15

0	Successful request; the resource has been unlocked.
4	The resource is not locked for the unlocking task.
8	TL format error.

The UNLOCK ALL macro does not provide a return code; register 15 remains unchanged.

VIRTAD

Name	Operation	Operand
[name]	VIRTAD	{address|(1)}

In S/370 mode, the VIRTAD macro returns the virtual address corresponding to a specified real address.

address|(1): Is the real storage address to be converted. It can be given as a symbol or in register notation.

In ECPS:VSE mode, only a virtual address can be specified as input. The macro returns that same address.

Register 0 returns the virtual address only if:

- for S/370 mode, the specified real address points to a page frame that contains a PFIXed page,
- for ECPS:VSE mode, the specified virtual address points to a PFIXed page.

Otherwise register 0 contains 0. Thus, the macro can be used to test if a page is PFIXed.

Note: The pages of a program running in real mode are considered to be fixed.

WAIT (PIOCS)

```
------------------------------------------------------
Name         Operation     Operand
------------------------------------------------------
[name]       WAIT          {blockname|(1)}
------------------------------------------------------
```

Issue this macro whenever your program requires that an I/O operation (started by an EXCP macro) be completed before execution of the program continues.

With the WAIT macro a task sets itself into the wait state until the specified control block (CCB or IORB) is posted (bit 0 of byte 2 turned on). Control blocks are normally used to synchronize tasks within the same partition. Do not use the WAIT macro for waiting on a CCB or IORB other than the one associated with the task, or with an I/O operation started by the same task. (For a description of the CCB or IORB see the corresponding macros).

When WAIT is processed, and the corresponding control block is not posted, the issuing task is set into the wait state. Control is then passed to the supervisor, which makes the processor available to another task in the same or in another partition.

The task issuing the WAIT macro remains in the wait state until the corresponding control block has been posted. In this case the event bit in the control block will be turned on.

blockname|(1): The blockname (specified as a symbol or in register notation) of the CCB or IORB established for the I/O device is the only operand required. This is also the same name as that specified in the EXCP macro for the device.

WAIT

```
-----------------------------------------------------
Name        Operation    Operand
-----------------------------------------------------
[name]      WAIT         {ecbname|(1)}
-----------------------------------------------------
```

With the WAIT macro a task sets itself into the wait state until the specified event control block (ECB, XECB, TECB, RCB, CCB, or IORB) is posted (bit 0 of byte 2 turned on). ECBs are normally used to synchronize tasks within the same partition. Use XECB support when tasks belong to different partitions. Do not use the WAIT macro for waiting on a telecommunication ECB, an RCB, a TECB, a CCB, or an IORB other than the one associated with the task (for instance, elapsed timer interval) or with an I/O operation started by the same task. (For a description of an ECB see the ATTACH macro. For a description of the RCB, TECB, CCB, or IORB see the corresponding macros.)

When WAIT is processed, and the corresponding event control block is not posted, the issuing task is set into the wait state. Control is then passed to the supervisor, which makes the processor available to another task in the same or in another partition.

The task issuing the WAIT macro remains in the wait state until the corresponding event control block has been posted. In this case the event bit in the event control block will be turned on.

> **Note:** When a wait is processed and the corresponding event bit is turned on, the task will keep control. If an ECB, TECB, or XECB is to be used more than once, it is the task's responsibility to reset the event bit as soon as possible after it has been posted.

WAITF

```
-----------------------------------------------------
Name         Operation    Operand
-----------------------------------------------------
[name]       WAITF        {filename1|(r1)}
                          [,filename2|,(r2)],...
-----------------------------------------------------
```

The WAITF macro is issued to ensure that the transfer of a record is complete. It is valid for both DAM and ISAM, but for SAM only with MICR and OCR devices. Filename is the same as that used in the DTF header entry, and may be specified either as a symbol or in register notation. Note that multiple filenames are valid only when using SAM to read MICR records.

The WAITF macro is issued after any READ or WRITE for a file and before the succeeding READ or WRITE for the same file. If the I/O operation is not completed when WAITF is issued, the active partition is placed in a wait state until the data transfer is completed. This allows processing of programs in other partitions while waiting for completion. When data transfer is complete, and if no errors were encountered, processing continues with the next sequential instruction. If an error is encountered, control passes to the error-handling routine provided for in the DTF.

If, however, you are using the multiple filename format of the WAITF macro while using MICR records, and if any of the files have records or errors ready to be processed, control remains in the partition and processing continues with the instruction following the WAITF.

WAITM

Name	Operation	Operand
[name]	WAITM	{ecb1,ecb2,...|listname|(1)}

The WAITM macro enables your program or task to wait for one of a number of events to occur. Control returns to the task when at least one of the event control blocks specified in the WAITM macro is posted. Refer to the WAIT macro for a description of the types of event control blocks and the restrictions on their usage.

The operand provides the addresses of the ECBs to be waited upon. The names of ecb1, ecb2... are assumed when at least two operands are supplied. A maximum of 16 names can be coded. If one operand is supplied, it is assumed to be the name (listname) of a list of consecutive full-word addresses that point to the ECBs to be waited upon. The first byte following the last address in the list must be nonzero to indicate the end of the list.

When control returns to a waiting task, register 1 points to the posted event control block (byte 2, bit 0 set on).

Note that a MICR CCB gets posted only when the device stops, not when a record is read. Furthermore, telecommunication ECBs and all RCBs must not be waited for, because their format does not satisfy a WAIT or a WAITM (that is, bit 0 of byte 2 would not be posted).

WRITE

```
---------------------------------------------------------------
Name      Operation   Operand
---------------------------------------------------------------
[name]    WRITE       {filename|(1)}
                      {,{SQ|UPDATE},{area|(0)}[,length|,(r)]
                       |,AFTER[,EOF]
                       |,ID
                       |,KEY
                       |,NEWKEY
                       |,RZERO}
---------------------------------------------------------------
```

The WRITE macro transfers a record from virtual storage to an output file.

filename|(1): Filename specifies the same name as that used in the DTF header entry. Register notation must be used if your program is to be self-relocating.

SQ|UPDATE: For sequential files, specify SQ for magnetic tape files. For disk work files, specify SQ for a formatting write and UPDATE for a non-formatting write.

When you are using control interval (CI) format, as with an FBA DASD, a non-formatting WRITE (with UPDATE) writes the current CI, while a formatting WRITE (with SQ) writes the CI and follows it immediately with a Software-End-Of-File (SEOF).

When not writing in CI format, as with CKD disk, a formatting WRITE writes count, key, and data, while a non-formatting WRITE writes only data.

area|(0): For sequential files, specifies the name, as a symbol or in register notation, of the output area used by the file.

length|(r): Specifies the actual number of bytes to be written on a sequential file. Is used only for records of undefined format (RECFORM=UNDEF).

AFTER: For DA files, specify AFTER to write a record after the last record written, regardless of key or identifier.

EOF: Is optional and applies only to the WRITE...AFTER form of the macro. Specify to write an end-of-file on a track after the last record on the track.

ID: For DA files, specify ID to write in a location determined by the record identifier in the count area of the records.

KEY: For indexed sequential files, specify KEY for random updating. For direct access files, specify KEY to write in a location determined by the record key (control information in the key area of the records).

NEWKEY: For indexed sequential files only; specify NEWKEY to write a new (not updated) record in the file. When loading or extending the file, precede the WRITE filename,NEWKEY with a SETFL macro and follow it with an ENDFL macro. When adding a record after sequential retrieval, issue an ESETL macro before writing the record.

RZERO: For DA files, specify RZERO to reset the capacity record of a track to its maximum value and erase the track after record zero.

XECBTAB

```
-----------------------------------------------------------
Name          Operation     Operand
-----------------------------------------------------------
[name]        XECBTAB       TYPE={DEFINE|DELETE|
                                  CHECK|RESET|DELETALL}
                            ,XECB=xecbname
                            [,XECBADR={xecbfield|
                                  (S,xecbfield)|(r1)}]
                            [,ACCESS={XPOST|XWAIT}]
                            [,MFG={area|(S,area)|(r2)}
-----------------------------------------------------------
```

The XECBTAB macro can be used

- to define, for the specified cross-partition event control block (XECB), an entry in the supervisor's XECB table,
- to delete such an entry,
- to check for the presence of an entry, or
- to reset an entry.

An XECB for which an entry has been defined to the supervisor can be referred to by XPOST and XWAIT macros; an XECB can be referred to also by a WAIT or WAITM macro if the task issuing the macro has previously defined the XECB (with ACCESS=XWAIT).

TYPE={DEFINE|DELETE|CHECK|RESET|DELETALL}: The operand specifies the type of operation to be performed:

DEFINE causes a new XECB table entry to be defined to the supervisor.

DELETE causes an entry to be deleted from the supervisor's XECB table. TYPE=DELETE can be specified only for an XECB for which an entry has been defined previously in the same program.

CHECK causes the system to check whether or not an entry for a specific XECB has been defined already. If that entry is present, specifying CHECK causes the address of both the XECB and the associated XECB table entry to be returned.

RESET causes the system to clear the information in the supervisor XECB table that indicates which task communicates with the program having defined the XECB. TYPE=RESET can be specified only for an XECB for which an entry has been previously defined in the same program.

After RESET, any task can attempt to establish a new connection with the owner. With ACCESS=XPOST, however, if the task currently connected to the XECB is issuing an XWAIT macro (at the time of RESET), this task will probably establish connection again, nullifying the RESET operation.

DELETALL performs three actions:

- Causes all entries in the XECB table that were defined previously by the issuing task to be deleted.
- Breaks the communication between any XECB owner and the issuing task (that is, clears the information in the XECB table that indicates that the issuing task communicates with the XECB owner).
- Posts all tasks as ready-to-run that are waiting for an XPOST by the issuing task. Also, the abnormal termination bit in the ECB is set on (bit 1 of byte 2). If these tasks are XWAITing on the XECB, they will get a return code of X'08' if they own the XECB.

Notes:

1. If TYPE=DELETALL is specified, the XECB and ACCESS operands must not be specified. XECBTAB with TYPE=DELETALL specified does not provide a return code; all registers remain unchanged.
2. XPOST partition abnormal termination is not indicated to tasks using WAIT or WAITM macro.

XECB=xecbname: Specifies the name of the XECB. If the XECBADR operand is not present, xecbname is the symbolic address of the 4-byte (or larger) XECB field. If, however, XECBADR is specified, xecbname is the name by which the control block is known between partitions; the symbolic address of the control block field is given by XECBADR.

The XECB field must be defined in your program, except when TYPE=CHECK is specified: in that case, the XECB field may be defined in another program.

XECBADR={xecbfield|(S,xecbfield)|(r1)}: XECBADR is used only if TYPE=DEFINE is specified. It provides the symbolic address of the 4-byte (or larger) field that is to be used as XECB.

ACCESS={XPOST|XWAIT}: This operand can be used together with TYPE=DEFINE to specify whether the program will be allowed to post the XECB or wait for another program to do the posting.

XPOST is assumed if the operand is omitted. It specifies that the program will be allowed to post the XECB. Specifying XPOST implies that only one other active task is allowed to issue an XWAIT macro against the XECB.

XWAIT specifies that the program will be allowed to wait for one other task to post the XECB.

MFG={area|(S,area)|(r2)}: The MFG operand is required if the program that issues the XECBTAB macro is to be reenterable. The operand specifies the address of a 64-byte storage area, that is, storage which your program obtains by a GETVIS macro. This area is needed for use by the system during execution of the macro.

The MFG operand is only useful in conjunction with XECBADR coded in either of the two notations (S,xecbfield) or (rl).

Feedback Information

Figure 31 shows the return codes that are supplied in register 15. The illustration also indicates whether or not the system returns the addresses of the pertinent XECB and the associated table entry in registers 1 and 14, respectively.

	X'00'	X'04'	X'08'
DEFINE	XECB named is stored in the table *	XECB named is already in the table **	The XECB table is full **
DELETE	XECB named is removed from the table **	XECB named was not found **	The issuing program did not define the XECB **
CHECK	XECB named was found in the table *	XECB named was not found **	N/A
RESET	XECB named communicat-ion bytes cleared **	XECB named was not found **	The issuing program did not define the XECB **

* Register 1 contains the address of the XECB and register 14 the address of the table entry.
** Registers 1 and 14 are set to zero.

Figure 31. XECBTAB Feedback Information

XPOST

```
---------------------------------------------------
Name         Operation    Operand
---------------------------------------------------
[name]       XPOST        XECB={xecbname|(1)}
                          POINTRG=(14)
---------------------------------------------------
```

The XPOST macro provides for cross-partition communication by posting the specified XECB (the macro sets bit 0 of byte 2 to 1). An XPOST macro issued against an XECB causes the task waiting for this XECB to be removed from the wait state (the waiting task may have issued an XWAIT, WAIT or WAITM with a previously defined XECB). This task may have been activated in the same or in another partition.

If the XPOST macro is used in a main-line loop, the macro should be preceded by a test which ensures that the other partition's task waiting for the event that is being posted must receive control and execute the function for which this event is a prerequisite.

To perform this test, a second XECB needs to be defined. This XECB allows the originally waiting task in its main-line loop to post completion of its function as an event for which the originally posting task must wait.

Resetting bit 0 of byte 2 of the XECB is a user responsibility.

Once a task has issued an XPOST macro for an XECB (with ACCESS=XWAIT), no other task can issue an XPOST for this XECB, until the connection is ended.

XECB={xecbname|(1)}: Specifies the name of the XECB to be posted. The name you specify must be the same as the one used to define the XECB. If register notation is used, the specified register must point to an 8-byte character field that contains the XECB name left-justified and padded with blanks. Do not specify 14 or 15 if you choose to use ordinary register notation.

POINTRG=(14): Specifies the register that points to the XECB table entry associated with the named XECB. Do not specify register 1 or 15 if you choose to use ordinary register notation.

To obtain the address of the associated XECB table entry, issue earlier in the program an XECBTAB macro for the same XECB and with TYPE=CHECK or TYPE=DEFINE specified. When the system executes the XECBTAB macro, it returns, in register 14, the address of the pertinent XECB table entry. Figure 5-8, which shows a coding example for the use of the XWAIT macro, applies to the XPOST macro accordingly.

Note that if the POINTRG register contains 0 (or any invalid value), all entries in the XECB table are searched to determine the correct address; no error is indicated.

Return Codes in Register 15

When the system returns control to the issuing task, register 15 contains one of the following return codes:

X'00'	Successful completion. The named XECB has been posted.
X'04'	The named XECB has no associated table entry in the XECB table.
X'0D'	The XECB referred to in the XPOST macro was defined with ACCESS=XPOST specified in the XECBTAB macro, but the task that issued the XPOST macro does not own this XECB.
X'0E'	The XECB referred to in the XPOST macro was defined with ACCESS=XWAIT specified in the XECBTAB macro and either (1) the task that issued the XPOST macro also defined the XECB or (2) the XECB has been posted previously during the same execution by another task.)

Note: Following the execution of an XPOST macro, registers 1 and 14 are set to zero.

```
-------------------------------------------------
Name        Operation    Operand
-------------------------------------------------
[name]      XWAIT        XECB={xecbname|(1)}
                         POINTRG=(14)
-------------------------------------------------
```

The XWAIT macro enables the issuing task to wait for an XECB to be posted by another task that is executing in the same or in another partition. Control returns to the issuing task when the XECB is posted or if an error condition is detected.

Once a task has issued an XWAIT macro for an XECB (with ACCESS=XPOST) to be posted, no other task can issue an XWAIT for this XECB, until the connection is ended.

XECB={xecbname|(1)}: Specifies the name of the XECB, which may be defined in the same or another program. The name you specify must be the same as the one used to define the XECB. If register notation is used, the specified register must point to an 8-byte field that contains the name of the XECB left-justified and padded with blanks. Do not specify register 14 or 15 if you choose to use ordinary register notation.

POINTRG=(14): Specifies the register that points to the XECB table entry associated with the named XECB. Do not specify register 1 or 15 if you choose to use ordinary register notation.

To obtain the address of the associated XECB table entry, issue earlier in the program an XECBTAB macro for the same XECB and with the TYPE=CHECK or TYPE=DEFINE specified. When the system executes the XECBTAB macro, it returns, in register 14, the address of the pertinent XECB table entry. Figure 32 on page 279 shows a coding example for the use of the XWAIT macro; in that example, the required continuation character is not shown. The example assumes that the XECB was defined by a program executing in another partition by (source) instructions as follows:

```
        .
        .
        .
        XECBTAB   TYPE=DEFINE,
                  XECB=MYECB
        .
        .
        .
MYECB   DC        F'0'
        .
        .
```

```
                .
WAITLP      XECBTAB  TYPE=CHECK,
                     XECB=MYECB
            LTR      15,15
            BNZ      ERROR
            LA       1,XECBNAME
            XWAIT    XECB=(1),
                     POINTRG=(14)
                .
                .
XECBNAME    DC       CL8'MYECB   '
```

Figure 32. Coding Example Showing the Use of XECBTAB with TYPE=CHECK and of XWAIT

Note that if the POINTRG register contains 0 (or any invalid value), all XECBs are searched to determine the correct address; no error is indicated.

Return Codes in Register 15

When the system returns control to the issuing task, register 15 contains one of the following return codes:

X'00'	Successful completion. The named XECB has been posted.
X'04'	The named XECB has no associated table entry in the XECB table or the owner of the XECB issued a DELETALL.
X'08'	The other task using this XECB has broken communication without issuing an XPOST. The task issuing the XWAIT is owner of the XECB.
X'0D'	The XECB referred to in the XWAIT macro was defined with ACCESS=XWAIT specified in the XECBTAB macro, but the task that issued the XWAIT macro does not own this XECB.
X'0E'	The XECB referred to in the XWAIT macro was defined with ACCESS=XPOST specified in the XECBTAB macro and either (1) the task that issued the XWAIT macro also defined the XECB or (2) the task did not define the XECB, but another task is already waiting for the XECB to be posted.

Note: Following the execution of an XWAIT macro, registers 1 and 14 are set to zero.

APPENDIX A. CONTROL CHARACTER CODES

CTLCHR=ASA

If the ASA option is chosen, a control character must appear in each record. If the control character for the printer is not valid, a message is given and the job is canceled. If the control character for card devices other than the 2560, 5424, and 5425 is not V or W, the card is selected into stacker 1. The codes are listed in Figure 33 on page 282.

Code	Interpretation
blank	Space one line before printing*
0	Space two lines before printing
–	Space three lines before printing
+	Suppress space before printing
1	Skip to channel 1 before printing*
2	Skip to channel 2 before printing
3	Skip to channel 3 before printing
4	Skip to channel 4 before printing
5	Skip to channel 5 before printing
6	Skip to channel 6 before printing
7	Skip to channel 7 before printing
8	Skip to channel 8 before printing
9	Skip to channel 9 before printing
A	Skip to channel 10 before printing
B	Skip to channel 11 before printing
C	Skip to channel 12 before printing
V	Select stacker 1
W	Select stacker 2
X	Select stacker 3 (2560 and 5424/5425 DTFCD files only)
Y	Select stacker 4 (2560 and 5424/5425 DTFCD files only)
X	Select stacker 5 (2560 DTFCD files only)
For DTFDI files on 2560 and 5424/5425:	
V	Primary hopper: select stacker 1
W	Primary hopper: select stacker 2
V	Secondary hopper: select stacker 5 (on 2560)
V	Secondary hopper: select stacker 4 (on 5424/5425)
W	Secondary hopper: select stacker 3

* For 3525 print (not associated) files, either space one or skip to channel 1 must be used to print on the first line of a card. For 3525 print associated files, only space one must be used to print on the first line of a card.

Figure 33. ASA Control Characters

CTLCHR=YES

The control character for this option is the command-code portion of the CCW used in printing a line or spacing the forms. The control codes are listed in Figure 34 on page 283 and Figure 35 on page 284.

Hexadecimal Code	Punch Combination	Function
Stacker selection on 1442 and 2596:		
81	12,0,1	Select into stacker 1
C1	12,1	Select into stacker 2
Stacker selection on 2520:		
01	12,9,1	Select into stacker 1
41	12,0,9,1	Select into stacker 2
Stacker selection on 2540:		
01	12,9,1	Select into stacker 1
41	12,0,9,1	Select into stacker 2
81	12,0,1	Select into stacker 3
13	11,3,9	Primary hopper: select into stacker 1
23	0,3,9	Primary hopper: select into stacker 2
33	3,9	Primary hopper: select into stacker 3
43	12,0,3,9	Primary hopper: select into stacker 4
53	12,11,3,9	Primary hopper: select into stacker 5 (2560 only)
93	12,11,3	Secondary hopper: select into stacker 1
A3	11,0,3	Secondary hopper: select into stacker 2
B3	12,11,0,3	Secondary hopper: select into stacker 3
C3	12,3	Secondary hopper: select into stacker 4
D3	11,3	Secondary hopper: select into stacker 5 (2560 only)
Stacker selection on 3504, 3505, and 3525:		
01	12,9,1	Select into stacker 1
41	12,0,9,1	Select into stacker 2

Figure 34. Stacker Selection Codes

Hexadecimal Code	Punch Combination	Function
Printer control (except for 3525):		
01	12,9,1	Write (no automatic space)
09	12,9,8,1	Write and space 1 line after printing
11	11,9,1	Write and space 2 lines after printing
19	11,9,8,1	Write and space 3 lines after printing
89	12,0,9	Write and skip to channel 1 after printing
91	12,11,1	Write and skip to channel 2 after printing
99	12,11,9	Write and skip to channel 3 after printing
A1	11,0,1	Write and skip to channel 4 after printing
A9	11,0,9	Write and skip to channel 5 after printing
B1	12,11,0,1	Write and skip to channel 6 after printing
B9	12,11,0,9	Write and skip to channel 7 after printing
C1	12,1	Write and skip to channel 8 after printing
C9	12,9	Write and skip to channel 9 after printing
D1	11,1	Write and skip to channel 10 after printing
D9	11,9	Write and skip to channel 11 after printing
E1	11,0,9,1	Write and skip to channel 12 after printing
0B	12,9,8,3	Space 1 line immediately
13	11,9,3	Space 2 lines immediately
1B	11,9,8,3	Space 3 lines immediately
8B	12,0,8,3	Skip to channel 1 immediately
93	12,11,3	Skip to channel 2 immediately
9B	12,11,8,3	Skip to channel 3 immediately
A3	11,0,3	Skip to channel 4 immediately
AB	11,0,8,3	Skip to channel 5 immediately
B3	12,11,0,3	Skip to channel 6 immediately
BB	12,11,0,8,3	Skip to channel 7 immediately
C3	12,3	Skip to channel 8 immediately
CB	12,0,9,8,3	Skip to channel 9 immediately
D3	11,3	Skip to channel 10 immediately
DB	12,11,9,8,3	Skip to channel 11 immediately
E3	0,3	Skip to channel 12 immediately
03	12,9,3	No operation

Figure 35 (Part 1 of 2). Printer Control Codes

Hexadecimal Code	Punch Combination	Function
Printer control for 3525 with Print Feature:		
0D	12,5,8,9	Print on line 1
15	11,5,9	Print on line 2
1D	11,5,8,9	Print on line 3
25	0,5,9	Print on line 4
2D	0,5,8,9	Print on line 5
35	5,9	Print on line 6
3D	5,8,9	Print on line 7
45	12,0,5,9	Print on line 8
4D	12,5,8	Print on line 9
55	12,11,5,9	Print on line 10
5D	11,5,8	Print on line 11
65	11,0,5,9	Print on line 12
6D	0,5,8	Print on line 13
75	12,11,0,5,9	Print on line 14
7D	5,8	Print on line 15
85	12,0,5	Print on line 16
8D	12,0,5,8	Print on line 17
95	12,11,5	Print on line 18
9D	12,11,5,8	Print on line 19
A5	11,0,5	Print on line 20
AD	11,0,5,8	Print on line 21
B5	12,11,0,5	Print on line 22
BD	12,11,0,5,8	Print on line 23
C5	12,5	Print on line 24
CD	12,0,5,8,9	Print on line 25

Figure 35 (Part 2 of 2). Printer Control Codes

APPENDIX B. AMERICAN NATIONAL STANDARD CODE FOR INFORMATION INTERCHANGE (ASCII)

In addition to the EBCDIC mode, VSE accepts magnetic tape files written in ASCII, a 128-character 7-bit code. The high-order bit in this 8-bit environment is zero. ASCII is based on the specifications of the American National Standards Institute, Inc..

VSE processes ASCII files in EBCDIC with the help of two translate tables, which are loaded into the SVA. Using these tables, logical IOCS translates from ASCII to EBCDIC all data as it is read into the I/O area. For ASCII output, logical IOCS translates data from EBCDIC to ASCII just before writing the record.

Figure 36 on page 288 shows the relative bit positions of the ASCII character set. An ASCII character is described by its column/row position in the table. The columns across the top of the figure list the three high-order bits. The rows along the left side of the figure are the four low-order bits.

For example, the letter P in ASCII is under column 5 and row 0 and is described in ASCII notation as 5/0. ASCII 5/0 and EBCDIC X'50' represent the same binary configuration (B'01010000'). However, P graphically represents this configuration in ASCII and & in EBCDIC. ASCII notation is always expressed in decimal. For example, the ASCII Z is expressed as 5/10 (not 5/A).

For those EBCDIC characters that have no direct equivalent in ASCII, the substitute character (SUB) is provided during translation. See Figure 37 on page 289 for ASCII to EBCDIC correspondence.

> **Note:** If an EBCDIC file is translated into ASCII, and you translate back into EBCDIC, this substitute character may not receive the expected value.

Bits[3]				b_7 →	0	0	0	0	1	1	1	1
				b_6 →	0	0	1	1	0	0	1	1
				b_5 →	0	1	0	1	0	1	0	1
b_4 ↓	b_3 ↓	b_2 ↓	b_1 ↓	Column → / Row ↓	0	1	2	3	4	5	6	7
0	0	0	0	0	NUL	DLE	SP	0	@	P	`	p
0	0	0	1	1	SOH	DC1	![1]	1	A	Q	a	q
0	0	1	0	2	STX	DC2	"	2	B	R	b	r
0	0	1	1	3	ETX	DC3	#	3	C	S	c	s
0	1	0	0	4	EOT	DC4	$	4	D	T	d	t
0	1	0	1	5	ENQ	NAK	%	5	E	U	e	u
0	1	1	0	6	ACK	SYN	&	6	F	V	f	v
0	1	1	1	7	BEL	ETB	'	7	G	W	g	w
1	0	0	0	8	BS	CAN	(	8	H	X	h	x
1	0	0	1	9	HT	EM	)	9	I	Y	i	y
1	0	1	0	10	LF	SUB	*	:	J	Z	j	z
1	0	1	1	11	VT	ESC	+	;	K	[	k	¦
1	1	0	0	12	FF	FS	,	<	L	\	l	¦
1	1	0	1	13	CR	GS	-	=	M	]	m	¦
1	1	1	0	14	SO	RS	.	>	N	∧[2]	n	~
1	1	1	1	15	SI	US	/	?	O	_	o	DEL

[1] The graphic I (Logical OR) may also be used instead of ! (Exclamation Point).

[2] The graphic ¬ (Logical NOT) may also be used instead of ^ (Circumflex).

[3] The 7 bit ASCII code expands to 8 bits when in storage by adding a high order 0 bit.

Example: Pound sign (#) is represented by

b_7	b_6	b_5	b_4	b_3	b_2	b_1
0	0	1	0	0	1	1

Control Character Representations

NUL	Null	DLE	Data Link Escape (CC)
SOH	Start of Heading (CC)	DC1	Device Control 1
STX	Start of Text (CC)	DC2	Device Control 2
ETX	End of Text (CC)	DC3	Device Control 3
EOT	End of Transmission (CC)	DC4	Device Control 4
ENQ	Enquiry (CC)	NAK	Negative Acknowledge (CC)
ACK	Acknowledge (CC)	SYN	Synchronous Idle (CC)
BEL	Bell	ETB	End of Transmission Block (CC)
BS	Backspace (FE)	CAN	Cancel
HT	Horizontal Tabulation (FE)	EM	End of Medium
LF	Line Feed (FE)	SUB	Substitute
VT	Vertical Tabulation (FE)	ESC	Escape
FF	Form Feed (FE)	FS	File Separator (IS)
CR	Carriage Return (FE)	GS	Group Separator (IS)
SO	Shift Out	RS	Record Separator (IS)
SI	Shift In	US	Unit Separator (IS)
		DEL	Delete

(CC)	Communication Control
(FE)	Format Effector
(IS)	Information Separator

Special Graphic Characters

SP	Space	<	Less Than
!	Exclamation Point	=	Equals
I	Logical OR	>	Greater Than
"	Quotation Marks	?	Question Mark
#	Number Sign	@	Commercial At
$	Dollar Sign	[	Opening Bracket
%	Percent	\	Reverse Slant
&	Ampersand	]	Closing Bracket
'	Apostrophe	∧	Circumflex
(	Opening Parenthesis	¬	Logical NOT
)	Closing Parenthesis	_	Underline
*	Asterisk	`	Grave Accent
+	Plus	¦	Opening Brace
,	Comma	¦	Vertical Line (This graphic is stylized to distinguish it from Logical OR)
-	Hyphen (Minus)	¦	Closing Brace
.	Period (Decimal Point)	~	Tilde
/	Slant		
:	Colon		
;	Semicolon		

Figure 36. ASCII Character Set

ASCII					EBCDIC				
Character	Col	Row	Bit Pattern		Col (in Hex)	Row	Bit Pattern		Comments
NUL	0	0	0000	0000	0	0	0000	0000	
SOH	0	1	0000	0001	0	1	0000	0001	
STX	0	2	0000	0010	0	2	0000	0010	
ETX	0	3	0000	0011	0	3	0000	0011	
EOT	0	4	0000	0100	3	7	0011	0111	
ENQ	0	5	0000	0101	2	D	0010	1101	
ACK	0	6	0000	0110	2	E	0010	1110	
BEL	0	7	0000	0111	2	F	0010	1111	
BS	0	8	0000	1000	1	6	0001	0110	
HT	0	9	0000	1001	0	5	0000	0101	
LF	0	10	0000	1010	2	5	0010	0101	
VT	0	11	0000	1011	0	B	0000	1011	
FF	0	12	0000	1100	0	C	0000	1100	
CR	0	13	0000	1101	0	D	0000	1101	
SO	0	14	0000	1110	0	E	0000	1110	
SI	0	15	0000	1111	0	F	0000	1111	
DLE	1	0	0001	0000	1	0	0001	0000	
DC1	1	1	0001	0001	1	1	0001	0001	
DC2	1	2	0001	0010	1	2	0001	0010	
DC3	1	3	0001	0011	1	3	0001	0011	
DC4	1	4	0001	0100	3	C	0011	1100	
NAK	1	5	0001	0101	3	D	0011	1101	
SYN	1	6	0001	0110	3	2	0011	0010	
ETB	1	7	0001	0111	2	6	0010	0110	
CAN	1	8	0001	1000	1	8	0001	1000	
EM	1	9	0001	1001	1	9	0001	1001	
SUB	1	10	0001	1010	3	F	0011	1111	
ESC	1	11	0001	1011	2	7	0010	0111	
FS	1	12	0001	1100	1	C	0001	1100	
GS	1	13	0001	1101	1	D	0001	1101	
RS	1	14	0001	1110	1	E	0001	1110	
US	1	15	0001	1111	1	F	0001	1111	
SP	2	0	0010	0000	4	0	0100	0000	
| ①	2	1	0010	0001	4	F	0100	1111	Logical OR
"	2	2	0010	0010	7	F	0111	1111	
#	2	3	0010	0011	7	B	0111	1011	
$	2	4	0010	0100	5	B	0101	1011	
%	2	5	0010	0101	6	C	0110	1100	
&	2	6	0010	0110	5	0	0101	0000	
'	2	7	0010	0111	7	D	0111	1101	
(	2	8	0010	1000	4	D	0100	1101	
)	2	9	0010	1001	5	D	0101	1101	
*	2	10	0010	1010	5	C	0101	1100	
+	2	11	0010	1011	4	E	0100	1110	
,	2	12	0010	1100	6	B	0110	1011	
-	2	13	0010	1101	6	0	0110	0000	Hyphen, Minus
.	2	14	0010	1110	4	B	0100	1011	
/	2	15	0010	1111	6	1	0110	0001	
0	3	0	0011	0000	F	0	1111	0000	
1	3	1	0011	0001	F	1	1111	0001	
2	3	2	0011	0010	F	2	1111	0010	
3	3	3	0011	0011	F	3	1111	0011	
4	3	4	0011	0100	F	4	1111	0100	
5	3	5	0011	0101	F	5	1111	0101	
6	3	6	0011	0110	F	6	1111	0110	
7	3	7	0011	0111	F	7	1111	0111	
8	3	8	0011	1000	F	8	1111	1000	
9	3	9	0011	1001	F	9	1111	1001	
:	3	10	0011	1010	7	A	0111	1010	
;	3	11	0011	1011	5	E	0101	1110	
<	3	12	0011	1100	4	C	0100	1100	
=	3	13	0011	1101	7	E	0111	1110	
>	3	14	0011	1110	6	E	0110	1110	
?	3	15	0011	1111	6	F	0110	1111	

Figure 37 (Part 1 of 2). ASCII to EBCDIC Correspondence

ASCII				EBCDIC			
Character	Col	Row	Bit Pattern	Col (in Hex)	Row	Bit Pattern	Comments
@	4	0	0100 0000	7	C	0111 1100	
A	4	1	0100 0001	C	1	1100 0001	
B	4	2	0100 0010	C	2	1100 0010	
C	4	3	0100 0011	C	3	1100 0011	
D	4	4	0100 0100	C	4	1100 0100	
E	4	5	0100 0101	C	5	1100 0101	
F	4	6	0100 0110	C	6	1100 0110	
G	4	7	0100 0111	C	7	1100 0111	
H	4	8	0100 1000	C	8	1100 1000	
I	4	9	0100 1001	C	9	1100 1001	
J	4	10	0100 1010	D	1	1101 0001	
K	4	11	0100 1011	D	2	1101 0010	
L	4	12	0100 1100	D	3	1101 0011	
M	4	13	0100 1101	D	4	1101 0100	
N	4	14	0100 1110	D	5	1101 0101	
O	4	15	0100 1111	D	6	1101 0110	
P	5	0	0101 0000	D	7	1101 0111	
Q	5	1	0101 0001	D	8	1101 1000	
R	5	2	0101 0010	D	9	1101 1001	
S	5	3	0101 0011	E	2	1110 0010	
T	5	4	0101 0100	E	3	1110 0011	
U	5	5	0101 0101	E	4	1110 0100	
V	5	6	0101 0110	E	5	1110 0101	
W	5	7	0101 0111	E	6	1110 0110	
X	5	8	0101 1000	E	7	1110 0111	
Y	5	9	0101 1001	E	8	1110 1000	
Z	5	10	0101 1010	E	9	1110 1001	
[	5	11	0101 1011	4	A	0100 1010	
\	5	12	0101 1100	E	0	1110 0000	Reverse Slant
]	5	13	0101 1101	5	A	0101 1010	
¬ [2]	5	14	0101 1110	5	F	0101 1111	Logical NOT
_	5	15	0101 1111	6	D	0110 1101	Underscore
`	6	0	0110 0000	7	9	0111 1001	Grave Accent
a	6	1	0110 0001	8	1	1000 0001	
b	6	2	0110 0010	8	2	1000 0010	
c	6	3	0110 0011	8	3	1000 0011	
d	6	4	0110 0100	8	4	1000 0100	
e	6	5	0110 0101	8	5	1000 0101	
f	6	6	0110 0110	8	6	1000 0110	
g	6	7	0110 0111	8	7	1000 0111	
h	6	8	0110 1000	8	8	1000 1000	
i	6	9	0110 1001	8	9	1000 1001	
j	6	10	0110 1010	9	1	1001 0001	
k	6	11	0110 1011	9	2	1001 0010	
l	6	12	0110 1100	9	3	1001 0011	
m	6	13	0110 1101	9	4	1001 0100	
n	6	14	0110 1110	9	5	1001 0101	
o	6	15	0110 1111	9	6	1001 0110	
p	7	0	0111 0000	9	7	1001 0111	
q	7	1	0111 0001	9	8	1001 1000	
r	7	2	0111 0010	9	9	1001 1001	
s	7	3	0111 0011	A	2	1010 0010	
t	7	4	0111 0100	A	3	1010 0011	
u	7	5	0111 0101	A	4	1010 0100	
v	7	6	0111 0110	A	5	1010 0101	
w	7	7	0111 0111	A	6	1010 0110	
x	7	8	0111 1000	A	7	1010 0111	
y	7	9	0111 1001	A	8	1010 1000	
z	7	10	0111 1010	A	9	1010 1001	
{	7	11	0111 1011	C	0	1100 0000	
¦	7	12	0111 1100	6	A	0110 1010	Vertical Line
}	7	13	0111 1101	D	0	1101 0000	
~	7	14	0111 1110	A	1	1010 0001	Tilde
DEL	7	15	0111 1111	0	7	0000 0111	

[1] The graphic ! (Exclamation Point) can be used instead of | (Logical OR).

[2] The graphic ^ (Circumflex) can be used instead of ¬ (Logical NOT).

Figure 37 (Part 2 of 2). ASCII to EBCDIC Correspondence

SC24-5211-1

VSE/Advanced Functions Application Programming: Macro Reference
(File No. S370/4300-30) Printed in U.S.A. SC24-5211-1

IBM®

READER'S COMMENT FORM

VSE/Advanced Functions
Application Programming: Macro Reference
Order No. SC24-5211-1

This form may be used to communicate your views about this publication. They will be sent to the author's department for whatever review and action, if any, is deemed appropriate. Comments may be written in your own language; use of English is not required.

IBM may use or distribute any of the information you supply in any way it believes appropriate without incurring any obligation whatever. You may, of course, continue to use the information you supply.

Note: *Copies of IBM publications are not stocked at the location to which this form is addressed. Please direct any requests for copies of publications, or for assistance in using your IBM system, to your IBM representative or to the IBM branch office serving your locality.*

Possible topics for comment are:

Clarity Accuracy Completeness Organization Coding Retrieval Legibility

If you wish a reply, give your name and mailing address:

____

____

____

What is your occupation?______________________________

Number of latest Newsletter associated with this publication:______________________________

Thank you for your cooperation. No postage stamp is necessary if mailed in the U.S.A. (Elsewhere, an IBM office or representative will be happy to forward your comments or you may mail directly to the address in the Edition Notice on the back of the title page.)

Note: Staples can cause problems with automated mail sorting equipment.
Please use pressure sensitive or other gummed tape to seal this form.

SC24-5211-1

Reader's Comment Form

Fold and tape | Please Do Not Staple | Fold and tape

NO POSTAGE NECESSARY IF MAILED IN THE UNITED STATES

BUSINESS REPLY MAIL

FIRST CLASS PERMIT NO. 40 ARMONK, N.Y.

POSTAGE WILL BE PAID BY ADDRESSEE:

International Business Machines Corporation
Department 812 BP
1133 Westchester Avenue
White Plains, New York 10604

Fold and tape | Please Do Not Staple | Fold and tape

IBM®

Cut or Fold Along Line

VSE/Advanced Functions Application Programming: Macro Reference
(File No. S370/4300-30) Printed in U.S.A. SC24-5211-1

READER'S
COMMENT
FORM

VSE/Advanced Functions
Application Programming: Macro Reference
Order No. SC24-5211-1

This form may be used to communicate your views about this publication. They will be sent to the author's department for whatever review and action, if any, is deemed appropriate. Comments may be written in your own language; use of English is not required.

IBM may use or distribute any of the information you supply in any way it believes appropriate without incurring any obligation whatever. You may, of course, continue to use the information you supply.

Note: *Copies of IBM publications are not stocked at the location to which this form is addressed. Please direct any requests for copies of publications, or for assistance in using your IBM system, to your IBM representative or to the IBM branch office serving your locality.*

Possible topics for comment are:

Clarity Accuracy Completeness Organization Coding Retrieval Legibility

If you wish a reply, give your name and mailing address:

____

____

____

What is your occupation?______________________________

Number of latest Newsletter associated with this publication:____________________

Thank you for your cooperation. No postage stamp is necessary if mailed in the U.S.A. (Elsewhere, an IBM office or representative will be happy to forward your comments or you may mail directly to the address in the Edition Notice on the back of the title page.)

Note: Staples can cause problems with automated mail sorting equipment.
Please use pressure sensitive or other gummed tape to seal this form.

SC24-5211-1

Reader's Comment Form

Fold and tape　　Please Do Not Staple　　Fold and tape

NO POSTAGE NECESSARY IF MAILED IN THE UNITED STATES

BUSINESS REPLY MAIL

FIRST CLASS　PERMIT NO. 40　ARMONK, N.Y.

POSTAGE WILL BE PAID BY ADDRESSEE:

International Business Machines Corporation
Department 812 BP
1133 Westchester Avenue
White Plains, New York 10604

Fold and tape　　Please Do Not Staple　　Fold and tape

IBM®

Cut or Fold Along Line

VSE/Advanced Functions Application Programming: Macro Reference
(File No. S370/4300-30)　Printed in U.S.A.　SC24-5211-1

SC24-5211-1

VSE/Advanced Functions Application Programming: Macro Reference
(File No. S370/4300-30) Printed in U.S.A. SC24-5211-1

IBM®

SC24-5211-1